IN THE UNITED STATES DISTRICT COURT

FOR THE DISTRICT OF DELAWARE

CRS, LLC Plaintiff, v. TURNER BROADCASTING SYSTEM, INC. Defendant.	))))))))))	C.A. No. 08-127-GMS

OPENING BRIEF IN SUPPORT OF DEFENDANT TURNER BROADCASTING SYSTEM, INC.'S MOTION TO TRANSFER TO THE WESTERN DISTRICT OF WASHINGTON

John W. Shaw (No. 3362)
Karen E. Keller (No. 4489)
YOUNG CONWAY STARGATT & TAYLOR, LLP
The Brandywine Building
1000 West Street, 17th Floor
Wilmington, Delaware 19801
(302) 571-6600
jshaw@ycst.com

OF COUNSEL:

Inge Larish
KLARQUIST SPARKMAN, LLC
One Union Square
600 University Street, Ste 2950
Seattle, Washington 98101

TABLE OF CONTENTS

Page

TABLE OF AUTHORITIES

Cases **Page**

Docketed Cases

Statutes

DEFENDANT TURNER BROADCASTING SYSTEM, INC.'S BRIEF IN SUPPORT OF ITS MOTION TO TRANSFER

I. INTRODUCTION

This is an action for patent infringement. The Plaintiff, CRS, LLC ("CRS") is a Washington corporation based in Seattle, Washington. The Defendant, Turner Broadcasting System, Inc. ("TBS, Inc.") is a Georgia corporation with its principal place of business in Atlanta, Georgia. The technology of the patent was developed in Seattle, Washington in the Western District of Washington. The named inventors and other key witnesses continue to reside in the Seattle area, are not party witnesses, and are not subject to this Court's compulsory process. There are no witnesses, no documents and no connections to the Delaware forum.

Under these circumstances, TBS, Inc. respectfully submits that the convenience of the parties and witnesses and the interest of justice will be best served if the present patent dispute is litigated in the Western District of Washington. The interests of justice also favor litigating this action in Washington because Plaintiff initiated a concurrent action in the Western District of Washington at Seattle involving the same patent against another defendant, Valve Corporation. Indeed, the Plaintiff has filed at least three actions in the Western District of Washington asserting the same patent, and Plaintiff's predecessor has previously filed at least four actions asserting the same or a parent patent. There is also ongoing litigation filed by TGN, Inc., a subsidiary of TBS, Inc., seeking a declaratory judgment of noninfringement and invalidity on the same patent and use of the same process at issue in this lawsuit.

Because there is no rational connection to the District of Delaware, because key nonparty witnesses can be compelled to appear at trial in the Western District of Washington, and because the Western District of Washington will necessarily adjudicate many of the claims on the same

or similar technology as pending in this Court, Defendant TBS, Inc. respectfully requests that this Court transfer this case to the Western District of Washington pursuant to 28 U.S.C. § 1404(a) and submits this brief in support.

II. STATEMENT OF FACTS AND NATURE AND STAGE OF PROCEEDINGS

A. Plaintiff and the Litigation History of the '124 Patent.

Plaintiff CRS, LLC is a Washington state company based in Seattle that is in the business of technology licensing. *See* Ex. A, License Renewal and Annual Report of CRS, LLC filed December 5, 2007. Plaintiff CRS purchased United States Patent No. 6,073,124 ("'124 Patent") in 2003 out of bankruptcy proceedings for the former Network Commerce, Inc. company ("Network Commerce") during a pending patent infringement action against Microsoft in the Western District of Washington. The Bankruptcy asset sale orders indicates that Plaintiff CRS purchased the '124 Patent, the rights to any infringement claims thereto and the "Microsoft" claim from the then pending litigation initiated by Network Commerce against Microsoft. *See* Ex. B, *Network Commerce, Inc. v. Microsoft Corp.,* Order Approving Sale, dated February 9, 2003.

The Washington State corporate filings list three members for CRS: Mr. Reed Corry, Mr. Joseph Schocken, and Mr. Robert Rohde. *See* Ex. C, License Renewal and Annual Report of CRS, LLC filed November 29, 2006. Shortly after acquiring the rights to the '124 Patent and the ongoing lawsuit, Mr. Rohde appeared as counsel on behalf of CRS and moved for CRS to intervene in the lawsuit. *See* Ex. D, *Network Commerce, Inc. v. Microsoft Corp.,* Motion to Intervene, dated March 24, 2004. CRS also moved to substitute for Network Commerce. The Western District of Washington granted the motion to intervene. *See* Ex. E, *Network Commerce,*

Inc. v. Microsoft Corp., Order Granting Motion to Intervene, dated April 20, 2004. Shortly thereafter, the Court granted Microsoft's motion for summary judgment of noninfringement of the '124 Patent. *See Network Commerce*, 260 F. Supp. 2d 1042, 1046-47 (W.D. Wash. 2003). As noted above, the judgment of noninfringement was affirmed by the Federal Circuit on appeal. *See Network Commerce*, 422 F.3d 1353 (Fed. Cir. 2005).

CRS has filed at least three patent infringement lawsuits asserting the '124 Patent in the Western District of Washington. *See CRS, LLC v. Bitarts* (W.D. Wash. 2:05-cv-0437-RSL); *CRS, LLC v. IGN* (W.D. Wash. 2:07-cv-00878-CMP); *CRS, LLC v. Valve* (W.D. Wash. 2:08-cv-00361-RAJ). Upon information and belief, its predecessor-in-interest, Network Commerce, had previously filed at least four patent infringement actions asserting either the '124 patent or its parent patent, United States Patent No. 6,141,698 ("the '698 Patent"), in the Western District of Washington. *See Network Commerce, Inc. v. Preview Sys.* (W.D. Wash. 2:00-cv-01790); *Network Commerce, Inc. v. Liquid Audio* (W.D. Wash. 2:01-cv-01540); *Network Commerce, Inc. v. Microsoft Corp.* (W.D. Wash. 2:01-cv-01991); *Network Commerce, Inc. v. Rainbow Tech.*, (W.D. Wash. 2:01-cv-01504). Upon information and belief, this litigation is the first time CRS or its predecessor has asserted the '124 Patent outside of the Western District of Washington.[1] Coppola Decl. ¶9.

Most recently, on March 3, 2008 – the same day as filing this matter – Plaintiff CRS filed Case No. 2:08-cv-00361 in the Western District of Washington asserting the '124 Patent against Valve Corporation ("Valve litigation") and accusing Valve's Steam website of allegedly

[1] TBS, Inc. has recently learned that CRS filed a second suit in Delaware, *CRS v. Reflexive Entertainment*. *See* C.A. No. 08-cv-00200-SLR. Except for the recently filed Reflexive case, TBS, Inc. is unaware of any other patent lawsuit filed by Plaintiff or Network Commerce outside of the Western District of Washington. Coppola Decl. ¶12.

infringing the '124 Patent. Valve filed its answer in the Western District of Washington on April 24, 2008. The accused Valve website may be used, among other things, to purchase computer games over the internet. Ex. F, *CRS, LLC v. Valve Corporation*, Dkt No. 13, Defendant Valve Corporation's Answer and Counterclaim.

On March 3, 2008, Plaintiff filed suit against Defendant TBS, Inc. in Delaware – with no apparent rationale for filing in this forum. TBS, Inc. is a Georgia Corporation, based in Atlanta, Georgia. Vigilante Decl. ¶6. TBS, Inc. has no offices or operational presence in Delaware. Vigilante Decl. ¶4. The Complaint alleges that use of the GameTap website allegedly infringes the '124 Patent. GameTap is a videogame website that is maintained by and on behalf of TGN, Inc., a subsidiary of TBS, Inc. TGN is also a Georgia corporation with no employees, property, or registered agent for service of process in Delaware. Vigilante Decl. ¶2. Among other things, GameTap may be used to purchase computer games over the internet. TBS, Inc. filed its answer in this Court on April 25, 2008, one day after Valve, alleging many of the same invalidity defenses as asserted in the Valve litigation. *See* Ex. F, *CRS, LLC v. Valve Corporation*, Dkt No. 13, Defendant Valve Corporation's Answer and Counterclaim.

On April 30, 2008, TGN, Inc.("TGN") filed suit against CRS in the Western District of Washington seeking a declaratory judgment on the invalidity of the '124 Patent and noninfringement of GameTap. The TGN lawsuit is currently pending before The Honorable Judge Marsha Pechman, the Judge that previously construed the claims of the '124 Patent and has substantial experience with the technology of the '124 Patent. *See Network Commerce v. Microsoft Corp.*, 260 F. Supp. 2d 1034, 1038-1041 (W.D. Wash. 2003); *Network Commerce*, 260 F. Supp. 2d at 1046-47, *aff'd Network Commerce*, 422 F.3d at 1363.

B. The '124 Patent.

The '124 Patent which is titled "Method and System for Securely Incorporating Electronic Information Into An Online Purchasing Application," concerns the delivery, purchase, and licensing of electronic merchandise (such as software or audio or video files) over a network such as the internet. The individuals listed as inventors of the '124 Patent are Ganapathy Krishnan, John Guthrie, and Scott Oyler. *See* Ex. G, U.S. Patent No. 6,073,124. Per the statements on the face of the patent, all three individuals lived in the Seattle area at the time the invention was developed. *Id.* All three initially assigned their alleged inventions to the predecessor of Network Commerce, Inc., ShopNow.com. ShopNow.com was also based in the Seattle, Washington area. Coppola Decl. ¶12. Upon information and belief, all three individuals currently reside in the Seattle area. Coppola Decl. ¶¶ 3-5.

Although the inventors had sought to obtain a much broader patent when they filed their original application in 1997, they had to narrow their claims significantly because prior art had already described numerous different systems for purchasing, delivering and licensing electronic merchandise over a network. Additionally, the application for the '124 Patent was a continuation-in-part of an application earlier in 1997, application no. 08/792,719 ("'719 application") which issued as U.S. Patent No. 6,141,698, filed in the name of only two of the three inventors eventually named on the '124 Patent, Ganapathy Krishnan and Scott Oyler. *See* Ex. H, U.S. Patent No. 6,141,698. The '124 Patent is a continuation-in-part of the parent patent. *See* Ex. G, U.S. Patent No. 6,073,124. Thus, in adding the additional inventor on the '124 Patent, the inventors also claimed new matter. The priority date of any given claim in the '124 Patent thus will depend, in part, on whether it contains the new matter added. *See* Ex. J, Manual

of Patent Examination and Procedure § 201.11, at 200-59, 60 (8th ed. 2007, incorporating rev. 6).

III. SUMMARY OF ARGUMENT

1. Transferring this case to the Western District of Washington will serve the interests of justice and judicial economy because the Western District of Washington has substantial experience with the '124 Patent and because Plaintiff affirmatively chose to file a concurrent litigation in the Western District of Washington which will require the Western District to consider many, if not most of the issues pertaining to the validity and claim construction that will be before this Court. Additionally, the case filed by TGN, Inc. in the Western District of Washington which will involve many of the same issues relating to the '124 Patent is currently pending before Judge Marsha Pechman, who has previously construed the claims of the '124 Patent and is familiar with the technology thereof. The efficiency that can be gained by leveraging the knowledge and experience of the Western District of Washington warrants transfer.

2. Plaintiff CRS' choice of forum is more easily overcome because this lawsuit has no apparent rational relationship to the state of Delaware, and Defendant is requesting transfer to Plaintiff's home turf and the location of the development of the technology of the '124 Patent.

3. Transferring this case to the Western District of Washington will also serve the convenience of the parties and witnesses. Delaware is not convenient for any of the parties. No documents or witnesses are located in Delaware. Plaintiff is located in the Western District of Washington and many key non-party witnesses are located beyond the subpoena power of this Court in the Western District of Washington.

IV. ARGUMENT

A. Legal Standards.

Congress designed the transfer statute, 28 U.S.C. 1404(a), "'to prevent the waste of time, energy and money' and 'to protect litigants, witnesses and the public against unnecessary inconvenience and expense.'" *Van Dusen v. Barrack*, 376 U.S. 612, 616 (1964), quoting *Continental Grain Co. v. Barge FBL*-585, 364 U.S. 19, 26 (1960): *see also Virgin Wireless, Inc. v. Virgin Enters, Ltd.*, 201 F. Supp. 2d 294, 299 (D. Del. 2002).

Application of the statute is committed to the sound discretion of the Court. *EEOC v. Univ. Pennsylvania*, 850 F.2d 969, 972, 977 (3d Cir. 1988), *aff'd*, 493 U.S. 182 (1990). Deference to the plaintiff's choice of forum does not trump the Congressional intent behind Section 1404(a) that federal litigation proceed in the district best suited to the interests of justice and convenience of the parties and witnesses. *See Plum Tree, Inc. v. Stockment*, 488 F.2d 754, 757-58 (3d Cir. 1973) (noting that the three Congressionally-articulated factors cannot be "automatically outweighed" by the parties' choice of forum); *Stewart Org., Inc. v. Ricoh Corp.*, 810 F.2d 1066, 1076 (11th Cir. 1987) (Tjoflat, J. concurring) ("In enacting section 1404(a), Congress instructed the federal courts to transfer civil actions whenever they deem transfer appropriate."), *aff'd*, 487 U.S. 22 (1988).

The Court is to "determine, on an individualized . . . basis, whether the convenience and fairness considerations weigh in favor of transfer." *Jumara v. State Farm Ins. Co.*, 55 F.3d 873, 883 (3d Cir. 1995) (citations omitted). To make this determination, the Court must "examine 'all relevant factors to determine whether, on balance, the litigation would more conveniently proceed and the interests of justice [would] better be served by a transfer to a different forum.'"

Affymetrix, Inc. v. Synteni, Inc., 28 F. Supp. 2d 192, 196-97 (D. Del. 1998) (citation omitted). The relevant factors include both private and public interests. *Id.* at 197. The private interests include: (1) the convenience of the expected witnesses; (2) the convenience of the parties; (3) the location of records and other documents; (4) the plaintiff's choice of forum and defendant's preferred forum; and (5) whether the claim arose elsewhere. The public interests include: (1) the practical considerations making trial easy, expeditious, or inexpensive; (2) administrative difficulties posed by the relative congestion of the two dockets in the respective fora; (3) any local interest in deciding local controversies at home; (4) the ability of the court to enforce the judgment. *Affymetrix,* 28 F. Supp. 2d at 197; *see also Jumara,* 55 F.3d at 879.

Although TBS, Inc. bears the burden of overcoming CRS' choice of forum and showing that this Court should transfer, *see e.g., Waste Distillation Tech., Inc. v. Pan Am. Res., Inc.*, 775 F. Supp. 759, 764 (D. Del. 1991), proper analysis of these factors here leads to the conclusion that this case should be transferred to the Western District of Washington, the Plaintiff's home turf, as demonstrated below.

B. The Private Interests Strongly Weigh In Favor of Transfer.

1. <u>Convenience of the Witnesses Strongly Supports Transfer.</u>

This Court has considered the ability to compel the attendance at trial of non-party witnesses an important, if not key, factor in the transfer inquiry. *See Mentor Graphics Corp. v. Quickturn Design Sys.*, 77 F. Supp. 2d 505, 510 (D. Del. 1999) ("The convenience of witnesses is often an important factor in a transfer inquiry.") (citing 15 Wright, Miller & Cooper, *Federal Practice and Procedure: Jurisdiction and Related Matters* § 3851, at 415 (2d ed. 1986)); *Affymetrix,* 28 F. Supp. 2d at 205 ("It is desirable to hold trial at a place where the personal

attendance of witnesses through the use of subpoena power can be reasonably assured.") (quoting *Pennwalt Corp. v. Purex Indus., Inc.*, 659 F. Supp. 287, 291 (D. Del. 1986)); *see also 3COM Corp. v. D-Link*, No. 03-014-GMS, Memorandum and Order, at 3-4 (D. Del April 25, 2003) (Ex K).

Absent a transfer in this case, Defendant will not be able to compel the attendance of key witnesses located in the Western District of Washington - Seattle area at trial in Delaware. Significantly, all three named inventors, Ganapathy Krishnan, John Guthrie, and Scott Oyler, reside in the Seattle area. Coppola Decl. ¶¶3-5. All three were identified in previous litigation by Network Commerce and CRS as "[t]he persons most knowledgeable concerning the conception and reduction to practice of the claimed invention". *See* Larish Decl. Ex. A, *Network Commerce, Inc. v. Microsoft Corp.*, Network Commerce Answers to Microsoft's First Set of Interrogatories Response to Interrogatory No. 2. There is a heightened likelihood that the inventors' testimony will be required because, as noted above, the '124 Patent claims priority from a previously filed application to which new material was added. Upon information and belief, Defendant further believes that one or more of the inventors' pre-filing activities may have invalidated the patent. Their testimony (and the testimony of other former employees) is important for this reason as well.

All three of the named inventors are not party employees. Thus, all three inventors are not subject to this Court's compulsory process. These key witnesses are also likely to be adverse to voluntarily appearing at trial in Delaware to assist Defendant in its invalidity defense and case against the patent which allegedly embodies their invention.

Mr. Dwayne Walker, who also resides in Seattle, is the former President of Network Commerce and the president of its predecessor company. Coppola Decl. ¶14 Upon information and belief, he was instrumental in directing research regarding technology associated with the '124 Patent, and its parent, the '698 Patent. Upon information and belief, Mr. Walker is thus a fact witness regarding the early development of the invention of the '124 Patent, the priority date, and other invalidity issues. Mr. Walker was also identified as a key witness in at least one set of initial disclosures in prior litigation by Network Commerce/CRS in the Western District of Washington. *See* Larish Decl. Ex. B, *Network Commerce, Inc. v. Microsoft Corp.*, Network Commerce Initial Disclosures. Mr. Walker is a not a party witness and not subject to the compulsory process of this Court.

Defendant may need to call other former employees of the now defunct Network Commerce company located in Seattle as fact witnesses as well. *See id.* These individuals are likely to be knowledgeable regarding the development of the invention, key dates, prior art and Defendant's invalidity defenses.

Finally, because the Seattle area was and remains a central location for the development of the technology of the '124 Patent, it is likely that numerous other witnesses will be located in the Western District of Washington.

Given the importance and potential number of non-party witnesses in the Western District of Washington, this factor weighs heavily **in favor of a transfer** to the Western District of Washington. *See 3COM Corp. v. D-Link,* No. 03-014-GMS, Ex. K at 4. ("At least two witnesses with knowledge of allegedly invalidating prior art are subject to compulsory process in northern California, but not Delaware. Even if these two witnesses were willing to travel to

Delaware to testify in this Court, it is certainly very inconvenient for them to do so, especially compared to traveling to a court in the state of their residence and employment.").

2. Location of Documents Favors Transfer.

No documents related to the case are known to exist in Delaware. Upon information and belief, many of the documents needed by Defendant for its defense are in Seattle.

When CRS purchased the '124 Patent out of the Network Commerce bankruptcy, it also purchased the rights to the then ongoing lawsuit between the (former) Network Commerce company and Microsoft (as noted above, this litigation was eventually resolved in favor of Microsoft on noninfringement grounds). Ex. I, Asset Purchase Agreement dated November 24, 2003. Mr. Rohde, the attorney appearing for CRS in the Microsoft action, is the same attorney as is representing CRS in this matter. Upon information and belief, CRS has the documents from Network Commerce relevant to the '124 Patent, along with documents related to the Network Commerce v. Microsoft litigation. Thus, these documents should be in the Western District of Washington at Seattle.[2]

TBS, Inc. will also need to inquire whether documentation still exists with individuals involved in the development and maintenance of the systems related to the invention of the '124 Patent. These documents would also be in Seattle to the best of TBS, Inc.'s knowledge and belief.

[2] Defendant TBS, Inc.'s counsel has documents that Network Commerce produced in the litigation with Microsoft and which were archived as permitted by the protective order in *Network Commerce v. Microsoft Corp.* TBS, Inc.'s counsel wrote to Plaintiff's counsel on April 11, 2008 to ascertain whether Plaintiff and its counsel have maintained Network Commerce documents and to request permission pursuant to the protective order entered in *Network Commerce v. Microsoft Corp.* to use the documents in the current litigation. As of the filing of this motion, TBS, Inc. has not received a response. Larish Decl. ¶19.

Because TBS, Inc. will likely be required to go to multiple non-party Seattle sources for documents, the location of the documents **favors transfer** to the Western District of Washington at Seattle.

3. CRS' Choice of Forum is More Easily Overcome Because There is No Rational and Legitimate Reason to Litigate in This Forum and Because Transfer is To CRS' Home Turf.

a) No Rational Relationship Exists With the Delaware Forum.

Although the plaintiff's choice of forum is entitled to deference in this inquiry, and should not be lightly disturbed, *Shutte v. Armco Steel Corp.,* 431 F.2d 22, 25 (3d Cir. 1920), when the plaintiff lacks a rational and legitimate reason to litigate in the forum, the transfer of a case to a more appropriate forum is less inconvenient. *See Sumito Mitsubishi Silicon Corp. v. MEMC Elec. Materials, Inc.*, 2005 WL 735880 (D. Del. 2005); *Waste Distillation Tech., Inc. v. Pan Am Res., Inc.*, 775 F. Supp. 759, 764 (D. Del. 1991); *Cont'l Cas. Co. v. Am. Home Assurance Co.*, 61 F. Supp. 2d 128, 131 (D. Del. 1999); *see also See Martin-Trigona v. Meister*, 668 F. Supp. 1, 2 (D.D.C. 1987); *Green Isle Partners, Ltd. v. The Ritz-Carlton Hotel, Co.*, No. 01-202-JJF, Memorandum Order, at 4-5 (D. Del. November 2, 2001) (Ex. L).

Similarly, where a plaintiff selects a forum other than its "home turf," less deference is given to the plaintiff's choice of forum. *See generally, Affymetrix.* 28 F. Supp. 2d 192, 198-200 (D. Del. 1998); *see also Burroughs Wellcome Co. v. Giant Food, Inc.*, 392 F. Supp. 761, 763 (D. Del. 1975) (Stapleton, J.) ("Where the forum selected by plaintiff is connected neither with the plaintiff nor with the subject matter of the lawsuit, meeting the burden of showing sufficient inconvenience to tip the 'balance' of convenience 'strongly in favor of defendant' will ordinarily be less difficult."); Ex. L at 4-5.

In this case, there is no rational connection between Plaintiff CRS, Defendant TBS, Inc., this lawsuit and this district. TBS, Inc. has no office in Delaware. Vigilante Decl. ¶4. TBS, Inc. has no registered agent in Delaware. Vigilante Decl. ¶5. TBS, Inc. has no operational presence in Delaware. Vigilante Decl. ¶4. TBS, Inc. is not incorporated in Delaware and, in fact, is a Georgia corporation with its principal place of business in Atlanta, Georgia. Vigilante Decl. ¶6.

Similarly, Plaintiff CRS is not connected to this forum. Not only is Delaware not Plaintiff's "home turf," the location of the requested transfer, the Seattle division of the Western District of Washington, is. Plaintiff CRS is a Washington corporation that has its principal place of business in Seattle, Washington. Indeed, CRS previously intentionally and repeatedly chose the Western District of Washington.[3]

Because this lawsuit has no apparent rational relationship to the state of Delaware, and Defendant is requesting transfer to Plaintiff's home turf, Plaintiff's choice of forum is entitled to considerably less deference and transfer is appropriate. *See, e.g.*, *Bayer Bioscience N.V. v. Monsanto Co.*, C.A. No. 03-023 GMS, 2003 WL 1565864, at *2 (D. Del. March 25, 2003) (noting that "while the defendant is a Delaware entity, and should reasonably expect to litigate in this forum there is little connection between Delaware and this action or the parties."); *Allergan, Inc. v. Alcon Labs.*, C.A. No. 02-1682-GMS, 2003 U.S. WL 473380, at *2 (D. Del. Feb. 25, 2003) (granting transfer and noting "there is little connection between Delaware and this action or the parties.").

[3] CRS sued IGN in the Western District of Washington. *See CRS, LLC v. IGN* (W.D. Wash. 2:08-cv-00878-CMP). IGN is a California company which was incorporated in Delaware and thus presumably had some rationale reason to expect to litigate in Delaware. Coppola Decl. ¶10.

b) Plaintiff Affirmatively Chose To Create a Concurrent Pending Lawsuit in the Western District of Washington.

Finally, CRS chose to sue TBS, Inc. in Delaware on the same day that it filed a sister lawsuit on the same patent against another company, Valve, in the Western District of Washington. Thus, Plaintiff affirmatively choose to create a pending lawsuit that would likely invoke many of the same defenses of invalidity and require the court of its home turf, the Western District of Washington, to be considering the same patent, potentially construing the same patent claims, and vying for the resources of the Seattle-based non-party witnesses as this Court.

4. There Will Be No Prejudice or Delay if Transfer is Granted.

There is little possibility for prejudice here because TBS, Inc. makes this motion at the onset of this case. TBS, Inc. answered on April 25, 2008, a scheduling order has not been entered and discovery has not yet begun.

Plaintiff's time to trial in the Western District of Washington is also likely to be faster, if anything, in the Western District of Washington than in this District, with median times to trial being respectively 18 and 27 months. Larish Decl. Ex. C, U.S. District Court Judicial Caseload Profiles for Western District of Washington and the District of Delaware. Plaintiff should not be prejudiced by this potentially faster resolution of its claims particularly as under Rule 11, Plaintiff is required to have thoroughly investigated its claims of alleged infringement prior to initiating this litigation. *See e.g. Judin v. United States*, 110 F.3d 780, 784-85 (Fed. Cir. 1997) (Rule 11 "requires that the inquiry be undertaken *before* the suit is filed, not after. Defendants have no choice when served with a complaint if they wish to avoid a default. . . Rule 11 prohibits

imposing those costs upon a defendant absent a basis, well-grounded in fact, for bringing the suit."). Thus, it is unlikely that transfer of this case would prejudice this case.

This factor **favors transfer**.

C. The Public Interests Strongly Weigh In Favor of Transfer.

1. <u>The Western District of Washington Has Substantial Familiarity With The Patent-In-Suit.</u>

To date, the Western District of Washington has considered the patent-in-suit or its parent patent in at least six prior lawsuits, not counting the most recently filed case by Plaintiff and the TGN litigation. The Western District of Washington has construed the claims of the patent, and rendered a judgment of noninfringement. In particular, Judge Pechman of the Western District of Washington has gained substantial familiarity with the technology and claims of the '124 Patent. Judge Pechman has construed the claims of the '124 Patent. *See Network Commerce,* 260 F. Supp. 2d at 1038-1041; *Network Commerce,* 260 F. Supp. 2d at 1046-47, *aff'd Network Commerce,* 422 F.3d at 1363. The efficiency that can be gained by leveraging the knowledge and familiarity that Judge Pechman and the Western District of Washington have already accumulated warrants transfer of this case to the Western District of Washington. *See also Conopco, Inc. v. Pfizer, Inc. and Princeton Biomedical Corp.*, No. 01-308-JJF, Memorandum Order (D. Del. November 15, 2001). ("Judge Hayden's previous rulings on the parent patents enable her to confront any new issues that might arise under the derivative '660 patent without any duplication of effort.") (Ex. N); *Altera Corp. v. XiLinx, Inc.*, No. 95-242-JJF, Memorandum Opinion (D. Del. March 29, 1996) ("judicial economy is better served by transferring this case to the Northern District of California. The district court in California is more familiar with the

complex technologies, product structures and prior art involved as well as at least one of the patents.") (Ex. M).

2. Trying This Case In the Western District Along With the Valve and TGN Litigation on the Same Patent Will Be More Efficient and Less Expensive Because The Pending Western District Litigations Will Require a Substantial Overlap of Court Resources and Witnesses.

Plaintiff, through its own choice, filed a lawsuit on the same patent in the Western District of Washington. This pending case will require the Western District to evaluate many, if not all, of the validity defenses pending before this Court. As noted above, the validity issues will likely require reliance on the same non-party witnesses that will be required to appear in two forums.

Additionally, the TGN litigation involves the same patent and the activities concerning the GameTap website alleged to be infringing in this case. The TGN litigation will likely require the same witnesses, documents and resources as the Valve litigation, filed by Plaintiff in the Western District of Washington. The pending TGN litigation will also require the Western District to evaluate many if not all of the validity defenses pending before the Western District in the Valve litigation and before this Court.

While this Court is amply qualified to familiarize itself with the technology and patent issues, transfer is appropriate because it would be inefficient for two different courts to "get up to speed" on the same subject matter, the same witnesses, and many of the same legal issues.

3. The Case Handling Statistics for the Western District of Washington Indicate That it Can Resolve This Matter Quickly.

The next public interest factor, the administrative difficulties posed by the relative congestion of the dockets in the respective fora, weighs in favor of transfer. *See Soloman v.*

Cont'l Am. Life Ins. Co., 472 F.2d 1043, 1047 (3d Cir. 1973) (approving consultation of official statistics when evaluating the public interest). As noted above, the Judicial Caseload Profile District Court Statistics indicate that the time from filing to disposition between the two courts is 27 months in this Court and 18 months in the Western District of Washington. Larish Decl. Ex. C, U.S. District Court Judicial Caseload Profiles for Western District of Washington and the District of Delaware.

While this Court has proven its ability to handle a heavy caseload, the *Jumara* analysis requires consideration of this factor. *See, e.g., Am. Sensor Rx., Inc. v. Banner Pharmcaps, Inc.*, C.A. No. 06-1929, 2006 WL 2583450, at *7 (D.N.J. Sept. 6, 2006) (transferring case in part because Middle District of North Carolina has a less congested docket with a shorter median time to trial); *Providian Life and Health Ins. Co. v. Cuna Mut. Ins. Soc'y*, No. 96-cv-1797, 1996 WL 153212, at *4 (E.D. Pa. Mar. 29, 1996) (granting motion to transfer partly because Western District of Wisconsin had shorter median times to disposition and trial which would "promote the expeditious resolution of this case"). Here, consideration of this factor, **favors transfer.**

4. Local Interest in Adjudicating Local Disputes.

There are no local ties to the State of Delaware. The Western District of Washington, in contrast, has specific ties to this litigation. The technology that is the stated subject of the patent was developed in the Western District of Washington. The inventors of the patent all reside in the Western District of Washington. The Plaintiff who is the alleged current patent owner resides in the Western District of Washington. Thus, this factor weighs in **favor of the transfer** to the Western District of Washington.

The public interest factors therefore **favor transfer** to the Western District of Washington.

D. This Case Could Have Been Brought in the Western District of Washington.

In a patent infringement action, venue is proper with respect to a defendant corporation in any judicial district where it resides. 28 U.S.C. § 1391(a), (c)). A corporation is deemed to reside in any district in which it is subject to personal jurisdiction at the time the action is commenced. 28 U.S.C. § 1391(c). There is no issue that the Western District of Washington has jurisdiction over CRS which is incorporated in Washington and has its office in Seattle. Therefore, this case could have been brought in the Western District of Washington, making that Court a proper transferee forum. 28 U.S.C. § 1404(a).

V. CONCLUSION

For all the foregoing reasons, Defendant respectfully requests that this Court grant this motion and transfer this case to the United States District Court for the Western District of Washington.

YOUNG CONWAY STARGATT & TAYLOR, LLP

/s/ John W. Shaw
John W. Shaw (No. 3362)
jshaw@ycst.com
Karen E. Keller (No. 4489)
kkeller@ycst.com
The Brandywine Building
1000 West Street, 17th Floor
Wilmington, Delaware 19801
(302) 571-6600

Of COUNSEL:

Inge A. Larish (admitted pro hac vice)
inge.larisha@klarquist.com
KLARQUIST SPARKMAN, LLP
One Union Square
600 University Street, Suite 2950
Seattle, Washington 98101
Telephone: 206-264-2960
Facsimile: 206-624-2719

J. Christopher Carraway
chris.carraway@klarquist.com
KLARQUIST SPARKMAN, LLP
One World Trade Center
121 S.W. Salmon Street, Suite 1600
Portland, Oregon 97204
Telephone: 503-595-5300
Facsimile: 503-595-5301

Attorneys for Defendant Turner Broadcasting System, Inc.

DATED: May 9, 2008

CERTIFICATE OF SERVICE

I, Karen E. Keller, Esquire, hereby certify that on May 9, 2008, I caused to be electronically filed a true and correct copy of the foregoing document with the Clerk of the Court using CM/ECF, which will send notification that such filing is available for viewing and downloading to the following counsel of record:

Michael G. Busenkell, Esquire
Eckert Seamans Cherin & Mellott LLC
300 Delaware Avenue, Suite 1210
Wilmington, DE 19801

I further certify that on May 9, 2008, I caused a true and correct copy of the foregoing document to be served by e-mail and hand delivery on the above-listed counsel of record and on the following non-registered participants in the manner indicated:

BY E-MAIL

Robert Rohde, Esquire, Esquire [brohde@rohdelaw.com]
Rohde & Van Kempen PLLC
1001 Fourth Avenue, Suite 4050
Seattle, WA 98154-1000

YOUNG CONAWAY STARGATT
& TAYLOR, LLP

/s/Karen E. Keller
Karen E. Keller (No. 4489) [kkeller@ycst.com]
The Brandywine Building
1000 West Street, 17th Floor
Wilmington, Delaware 19899
(302) 571-6600

EXHIBIT A

UNITED STATES OF AMERICA

The State of Washington

Secretary of State

I, Sam Reed, Secretary of State of the State of Washington and custodian of its seal, hereby issue this

certificate that the attached is a true and correct copy of

LICENSE RENEWAL AND ANNUAL REPORT

of

CRS, LLC

as filed in this office on December 5, 2007.

Date: April 17, 2008

Given under my hand and the Seal of the State of Washington at Olympia, the State Capital

Sam Reed, Secretary of State

STATE OF WASHINGTON
DEPARTMENT OF LICENSING
MASTER LICENSE SERVICE
Renewal Agent for Secretary of State

[redacted]

Limited Liability Company License Renewal & Annual Report

Profit Corporation Name, Registered Agent, and Registered Office Address

FOR VALIDATION ONLY

01P-400-925-0003

0745-W

CRS, LLC
C/O JOSEPH SCHOCKEN
600 UNIVERSITY ST #2800
SEATTLE WA 98101

Unified Business ID No. 602 350 476
State of Formation WA
Date of Formation 12-22-2003
Expiration Date* 12-31-2007

☐ If the registered agent and/or office address shown above has changed, mark the box and complete the reverse side.

LICENSE RENEWAL SECTION *After renewal your new expiration date will be: 12-31-2008

RENEW ONLINE! Go to: www.dol.wa.gov/business/renewcorp.html Use your UBI# and the password: TB37 4672

Domestic Limited Liability Company Renewal Application Fee	$ 50.00
	9.00

Failure to return completed form and pay fees by the expiration date will result in $25.00 late fee and may lead to the dissolution of your company.

Make check payable to: STATE TREASURER in U.S. FUNDS only

TOTAL FEES DUE: $59.00

FEES & REPORT REQUESTED BY: 12-17-2007

ANNUAL REPORT SECTION – The *entire section* below must be completed *each year.* Type or print legibly in dark ink.

Does your company own land, buildings, or other real property in Washington? ☐ Yes ☒ No (If Yes, see instructions on reverse side under "Controlling Interest")

Contact telephone no. (206) 623-1700 Contact e-mail address jls@broadmark.com

Address of principal place of business 600 University Street #2800 Seattle WA 98101
ADDRESS CITY STATE ZIP

If formed outside Washington, list the LLC office address
ADDRESS CITY STATE ZIP

Briefly Describe the Nature of Your Business Technology Licensing
(Example: Retail sales. Stating "Any lawful purpose" is not adequate under Washington State law and will be rejected)

List title, name, and address of managers, if applicable. Otherwise list title, name, and address of members *(attach additional sheets in the same format, if necessary. Include your UBI number on each page).*

TITLE	NAME	ADDRESS	CITY	STATE	ZIP
manager	Joseph L. Schocken	600 University St #2800	Seattle	WA	98101

Is the Limited Liability Company managed by managers? ☒ Yes ☐ No

602 350 476

X [signature] manager 11/30/2007
SIGNATURE OF MEMBER OR MANAGER TITLE DATE SIGNED

FORM MUST BE SIGNED BY A MEMBER OR MANAGER LISTED ABOVE

Telephone: (360) 664-1450

Please return to: DEPARTMENT OF LICENSING
MASTER LICENSE SERVICE
PO BOX 9034
OLYMPIA WA 98507-9034

BLS-700-350 LLC RENEWAL (R/11/06)

LIMITED LIABILITY COMPANY CERTIFICATE OF [illegible]
REGISTERED AGENT OR REGISTERED OFFICE ADDRESS

No fee if filed in conjunction with the License Renewal and Annual Report

Please type or print legibly in dark ink.

1. Limited Liability Company Name: _____

   Unified Business Identifier Number: _____

2. Printed Name of New Registered Agent: _____

   The Registered Agent must be either an individual who is a resident of the state of Washington with a business address the same as the Registered Office address shown below; or a corporation, limited liability company *(different from this limited liability company)*, or limited partnership registered with the Washington Secretary of State to do business in Washington, and with a business office address the same as the Registered Office address entered below. The new agent, or its authorized representative, must also sign the consent to appointment below.

3. The registered office street address is required. It must be identical to the **business** address of the Registered Agent and *must be located in the state of Washington. A Post Office Box address may be used for mailing purposes only.*

   New Registered Address: _____ (REQUIRED: STREET & NUMBER OR RURAL ROUTE) _____ (CITY) **WA** _____

   PO Box for Mailing: _____ (OPTIONAL: POST OFFICE BOX NUMBER) _____ (CITY) **WA** _____ (ZIP)

4. Consent to Appointment as New Registered Agent

   I consent to serve as Registered Agent in the state of Washington for the above named limited liability company. As such, I understand that it will be my responsibility to accept Service of Process on behalf of the limited liability company, to forward mail to the limited liability company, and to immediately notify the Office of the Secretary of State if I resign or change the Registered Office Address.

X _____

SIGNATURE OF THE AGENT SHOWN ABOVE *(IF THE AGENT IS A CORPORATION, LIST YOUR CORPORATE TITLE AFTER SIGNATURE.)* DATE

CONTROLLING INTEREST

Answer the following question **only** if you answered "yes" to the question about owning land, buildings, or other real property in Washington on the front of this form:

Has there been a change of 50% or more of the ownership of stock or other interest in the company during the last 12 months? ☐ Yes ☐ No

You must contact the Washington State Department of Revenue about excise taxes **IF:**

- This company owns land, buildings, or other real estate in Washington State, **AND**
- 50% or more of the ownership ("controlling interest") in this company, such as ownership of stocks or other financial interests, changed hands during the past 12 months *(RCW 82.45.033)*.

Failure to report a change can be penalized *(RCW 82.32.090(6), 82.45.100)*

For more information on Controlling Interest, please call the Department of Revenue at (360) 570-3265.

The Department of Licensing has a policy of providing equal access to its services. If you need special accommodation, please call (360) 664-1400 or TTY (360) 664-8885.

EXHIBIT B

The Honorable Philip H. Brandt
Chapter 11
Response Date: December 5, 2003

UNITED STATES BANKRUPTCY COURT FOR THE
WESTERN DISTRICT OF WASHINGTON AT SEATTLE

In re: NETWORK COMMERCE INC., Debtor.	NO. 02-23396-PHB11 ORDER APPROVING SALE UNDER ORDER AUTHORIZING DEBTOR TO CONDUCT SALE OF PATENTS, MICROSOFT CLAIM, AND MICROSOFT CASE PURSUANT TO AUCTION AND SECTION 363(b), (f), AND (m) OF THE BANKRUPTCY CODE

THIS MATTER having come before the Court on the Debtor's Motion for Order Authorizing Debtor to Sell Patents, Microsoft Claim, and Microsoft Case to TechSearch, LLC Pursuant to Section 363(b), (f), and (m) of the Bankruptcy Code (the "Motion"); the Court having entered the Order Authorizing Debtor to Conduct Sale of Patents, Microsoft Claim, and Microsoft Case Pursuant to Auction and Section 363(b), (f), and (m) of the Bankruptcy Code on November 20, 2003 (the "Auction Order"), the definitions, terms, and conditions of which are incorporated herein and made applicable hereto by this reference; the Debtor having conducted

Cairncross & Hempelmann, P.S.
Law Offices
524 Second Avenue, Suite 500
Seattle, Washington 98104-2323
Phone: 206-587-0700 • Fax: 206-587-2308

the Auction pursuant to the terms of the Auction Order; the Debtor having filed a notice identifying the Successful Bidder as CRS, LLC, with a copy of the Agreement attached, pursuant to paragraph 2 of the Auction Order; the Court finding that the sale of the Assets is based on sound business justifications, and such sale is in the best interests of the Debtor's estate based on the results of the Auction and for the reasons set forth in the Motion, the Memorandum, and the Dickson Declaration; the Court further finding that the sale of the Assets pursuant to the Agreement has been proposed and, if consummated, will have been consummated in good faith in accordance with 11 U.S.C. § 363(m); the Court further finding that the Successful Bidder is a good faith purchaser and is entitled to the protections afforded under 11 U.S.C. § 363(m) and that if the transaction contemplated by the Agreement closes, the Successful Bidder will have acted in good faith in closing the transaction; the Court further finding that the Successful Bidder is not an insider or affiliate of the Debtor; the Court further finding that the consideration to be received by the Debtor from the Successful Bidder is fair and reasonable, and that the sale does not unfairly benefit insiders, a proprietary purchaser, or any creditor or class of creditors; and the Court further finding that consummation of the Agreement is in the best interests of the Debtor, its estate, all creditors, and other parties in interest,

NOW, THEREFORE, it is HEREBY ORDERED:

1. The Agreement is approved, and the Debtor is authorized to enter into the Agreement. A copy of the Agreement is attached hereto as Exhibit 1.

2. Except as otherwise set forth by this Order or the Agreement, in accordance with 11 U.S.C. § 363(f), the sale of the Assets pursuant to the Agreement is and shall be free and clear of any interest in such Assets of an entity other than the estate (whose ongoing interest with respect to the Assets shall be limited to the extent of the Purchase Price as expressly provided in

Cairncross & Hempelmann, P.S.
Law Offices
524 Second Avenue, Suite 500
Seattle, Washington 98104-2323
Phone: 206-587-0700 • Fax: 206-587-2308

the Agreement), including but not limited to all claims, liens, and encumbrances of any nature, kind, or description, with such interests, claims, liens, and encumbrances to attach to the sale proceeds in the order and priority that existed prior to the sale.

3. The Agreement was proposed, negotiated, and entered into in good faith after arm's length bargaining by the parties and the Auction and provides the Debtor with the highest or otherwise best offer received for the Assets. The Successful Bidder is a good faith purchaser pursuant to 11 U.S.C. § 363(m) and entitled to the protections thereunder.

4. The Debtor is authorized to take such further actions as may be necessary to implement, close, and consummate the sale of the Assets pursuant to the terms of the Agreement without further notice or order of this Court.

5. The Successful Bidder has not assumed or otherwise become obligated for any of the Debtor's liabilities. All creditors of the Debtor, whether known or unknown, are hereby enjoined from asserting or prosecuting a claim or cause of action against the Successful Bidder or the Assets to recover on account of any liability owed by the Debtor.

6. This Order shall not be subject to the stay or Federal Rule of Bankruptcy Procedure 6004(g).

7. The Successful Bidder shall pay the Purchase Price (as defined in the Agreement) pursuant to the terms of the Agreement.

8. The Successful Bidder shall not be deemed a successor to the Debtor, by reason of the Closing (as defined in the Agreement) or otherwise.

///

Cairncross & Hempelmann, P.S.
Law Offices
524 Second Avenue, Suite 500
Seattle, Washington 98104-2323
Phone: 206-587-0700 • Fax: 206-587-2308

9. Federal, state, and local governmental units are directed to accept any and all documents to effectuate the Closing.

DATED this 9 day of December, 2003.

The Honorable Philip H. Brandt
United States Bankruptcy Judge

Presented by:

Cairncross & Hempelmann, P.S.

/s/ John R. Rizzardi
John R. Rizzardi, WSBA # 9388

Attorneys for Debtor Network Commerce Inc.

Cairncross & Hempelmann, P.S.
Law Offices
524 Second Avenue, Suite 500
Seattle, Washington 98104-2323
Phone: 206-587-0700 • Fax: 206-587-2308

EXHIBIT C

UNITED STATES OF AMERICA

The State of Washington

Secretary of State

I, Sam Reed, Secretary of State of the State of Washington and custodian of its seal, hereby issue this

certificate that the attached is a true and correct copy of

LICENSE RENEWAL AND ANNUAL REPORT

of

CRS, LLC

as filed in this office on November 29, 2006.

Date: April 17, 2008

Given under my hand and the Seal of the State of Washington at Olympia, the State Capital

Sam Reed, Secretary of State

STATE OF WASHINGTON
DEPARTMENT OF LICENSING
MASTER LICENSE SERVICE
Renewal Agent for Secretary of State

[redacted]

Limited Liability Company License Renewal & Annual Report

Profit Corporation Name, Registered Agent, and Registered Office Address

FOR VALIDATION ONLY

5162 000 400 112906 59.00

01P-400-925-0003

0645-W

Unified Business ID No. 602 350 476

State of Formation WA

Date of Formation 12-22-2003

Expiration Date* 12-31-2006

CRS, LLC
C/O JOSEPH SCHOCKEN
600 UNIVERSITY ST #2800
SEATTLE WA 98101

☐ If the registered agent and/or office address shown above has changed, mark the box and complete the reverse side.

LICENSE RENEWAL SECTION *After renewal your new expiration date will be: 12-31-2007

RENEW ONLINE! Go to: www.dol.wa.gov/business/renewcorp.html Use your UBI# and the password: T5F2 8526

Domestic Limited Liability Company	$ 50.00
Renewal Application Fee	9.00

Failure to return completed form and pay fees by the expiration date will result in $25.00 late fee and may lead to the dissolution of your company.

Make check payable to: STATE TREASURER in U.S. FUNDS only

TOTAL FEES DUE: $59.00

FEES & REPORT REQUESTED BY: 12-15-2006

ANNUAL REPORT SECTION – The *entire section* below must be completed *each year.* Type or print legibly in dark ink.

Does your company own land, buildings, or other real property in Washington? ☐ Yes ☒ No (If Yes, see instructions on reverse side under "Controlling Interest")

Contact telephone no. (206) 623-1200 Contact e-mail address

Address of principal place of business 600 UNIVERSITY ST #2800 SEATTLE WA 98101-4123
(ADDRESS / CITY / STATE / ZIP)

If formed outside Washington, list the LLC office address
(ADDRESS / CITY / STATE / ZIP)

Briefly Describe the Nature of Your Business Technology - Licensing
(Example: Retail sales. Stating "Any lawful purpose" is not adequate under Washington State law and will be rejected)

List title, name, and address of managers, if applicable. Otherwise list title, name, and address of members *(attach additional sheets in the same format, if necessary. Include your UBI number on each page).*

TITLE	NAME	ADDRESS	CITY	STATE	ZIP
Member	Reed A Corry	600 University St #2800	Seattle	WA	98101
Member	Tranceks, LLC	600 University St #2800	Seattle	WA	98101
Member	Robert E. Rohde	1001 Fourth Ave # 4050	Seattle	WA	98154-1000

Is the Limited Liability Company managed by managers? ☒ Yes ☐ No

X [signature] Joseph L. Schocken Manager of Tranceks, LLC 602 350 476

SIGNATURE OF MEMBER OR MANAGER / TITLE / DATE SIGNED

FORM MUST BE SIGNED BY A MEMBER OR MANAGER LISTED ABOVE

Telephone: (360) 664-1450

Please return to: DEPARTMENT OF LICENSING
MASTER LICENSE SERVICE
PO BOX 9034
OLYMPIA WA 98507-9034

BLS-700-350 LLC RENEWAL (R/3/06)

LIMITED LIABILITY COMPANY CERTIFICATE OF CHANGE OF REGISTERED AGENT OR REGISTERED OFFICE ADDRESS

No fee if filed in conjunction with the License Renewal and Annual Report

Please type or print legibly in dark ink.

1. Limited Liability Company Name: _____

   Unified Business Identifier Number: _____

2. Printed Name of New Registered Agent: _____

   The Registered Agent must be either an individual who is a resident of the state of Washington with a business address the same as the Registered Office address shown below; or a corporation, limited liability company *(different from this limited liability company)*, or limited partnership registered with the Washington Secretary of State to do business in Washington, and with a business office address the same as the Registered Office address entered below. The new agent, or its authorized representative, must also sign the consent to appointment below.

3. The registered office street address is required. It must be identical to the **business** address of the Registered Agent and must be located in the state of Washington. A Post Office Box address may be used for mailing purposes only.

   New Registered Address: _____ (**REQUIRED:** STREET & NUMBER OR RURAL ROUTE) _____ (CITY) **WA** _____

   PO Box for Mailing: _____ (**OPTIONAL:** POST OFFICE BOX NUMBER) _____ (CITY) **WA** _____ (ZIP)

4. Consent to Appointment as New Registered Agent

   I consent to serve as Registered Agent in the state of Washington for the above named limited liability company. As such, I understand that it will be my responsibility to accept Service of Process on behalf of the limited liability company, to forward mail to the limited liability company, and to immediately notify the Office of the Secretary of State if I resign or change the Registered Office Address.

X _____

SIGNATURE OF THE AGENT SHOWN ABOVE *(IF THE AGENT IS A CORPORATION, LIST YOUR CORPORATE TITLE AFTER SIGNATURE.)* DATE

CONTROLLING INTEREST

Answer the following question **only** if you answered "yes" to the question about owning land, buildings, or other real property in Washington on the front of this form:

Has there been a change of 50% or more of the ownership of stock or other interest in the company during the last 12 months? ☐ Yes ☐ No

You must contact the Washington State Department of Revenue about excise taxes **IF:**

- This company owns land, buildings, or other real estate in Washington State, **AND**
- 50% or more of the ownership ("controlling interest") in this company, such as ownership of stocks or other financial interests, changed hands during the past 12 months *(RCW 82.45.033)*.

Failure to report a change can be penalized *(RCW 82.32.090(6), 82.45.100)*

For more information on Controlling Interest, please call the Department of Revenue at (360) 570-3265.

The Department of Licensing has a policy of providing equal access to its services. If you need special accommodation, please call (360) 664-1400 or TTY (360) 664-8885.

EXHIBIT D

THE HONORABLE MARSHA J. PECHMAN

01-CV-01991-M

UNITED STATES DISTRICT COURT
WESTERN DISTRICT OF WASHINGTON
AT SEATTLE

NETWORK COMMERCE, INC., a Washington corporation,

Plaintiff,

CRS, LLC, Applicant for Intervention,

v.

MICROSOFT CORPORATION, a Washington corporation,

Defendant.

NO. C01-1991P

MOTION OF CRS, LLC TO INTERVENE AS ADDITIONAL PLAINTIFF

NOTE ON MOTION CALENDAR: APRIL 9, 2004

INTRODUCTION

CRS, LLC has purchased all right, title and interest to the patents of Network Commerce, Inc. ("NCI") and Network Commerce Inc.'s patent infringement claim against Microsoft. CRS, LLC brings this motion to intervene to be added as a plaintiff in this matter under FRCP 24(a)(2).

BACKGROUND

By agreement dated November 2003, CRS, LLC agreed to purchase all right, title and interest to NCI's patents, including the '124 patent asserted here, as well as all right, title and interest to NCI's claim for patent infringement against Microsoft. Ex. A. The agreement was

ROHDE & VAN KAMPEN PLLC
1000 Second Avenue, Suite 3110
Seattle, Washington 98104-1046
(206) 386-7353

approved by the Bankruptcy Court on December 9, 2003 and the assignment of the '124 patent and NCI's claim against Microsoft was executed on January 21, 2004. Exs. B and C.

CRS clearly "claims an interest relating to the property or transaction which is the subject of the action." Moreover, NCI is in bankruptcy and has filed a plan under which it will be liquidated. Because NCI has no active management or operations nor the resources to pursue this claim against Microsoft, the "disposition of the action may as a practical matter impair or impede the applicant's ability to protect that interest." Accordingly, CRS, LLC respectfully requests that the Court add it as a plaintiff in this matter. A pleading setting forth the claim for which intervention is desired is attached. Ex. D.

DATED this 18th day of March, 2004.

ROHDE & VAN KAMPEN, PLLC

By: _____
Robert E. Rohde, WSBA #12809
Attorneys for CRS, LLC

ROHDE & VAN KAMPEN PLLC
1000 Second Avenue, Suite 3110
Seattle, Washington 98104-1046
(206) 386-7353

DECLARATION OF SERVICE

I hereby declare, under penalty of perjury under the laws of the State of Washington, that one copy of the attached MOTION OF CRS, LLC TO INTERVENE AS ADDITIONAL PLAINTIFF is being deposited with the United States Postal Service with sufficient postage as first-class mail in an envelope addressed to:

Ramsey M. Al-Salam
Perkins Coie
1201 – 3rd Avenue, Suite 4800
Seattle, WA 98101-3099

and to:

Mark S. Parris
Heller Ehrman White & McAuliffe LLP
701 – 5th Avenue, Suite 6100
Seattle, WA 98104-7098

and to:

J. Christopher Carraway
Joseph T. Jakubek
Klarquist Sparkman, LLP
One World Trade Center, Suite 1600
121 SW Salmon Street
Portland, OR 97204

and to:

Stephen P. McGrath
Microsoft Corporation
One Microsoft Way, Building 8
Redmond, WA 98052-6399

on March 19, 2004.

Dated this 19th day of March, 2004.

Jeff Nowlin

ROHDE & VAN KAMPEN PLLC
1000 Second Avenue, Suite 3110
Seattle, Washington 98104-1046
(206) 386-7353

Asset Purchase Agreement

This Asset Purchase Agreement (hereinafter "Agreement"), is entered into as of November 24, 2003 (the "Effective Date"), by and between Network Commerce Inc., a Chapter 11 debtor in possession, United States Bankruptcy Court for the Western District of Washington (the "Bankruptcy Court") (hereinafter "Seller"), and CRS, LLC, an Washington Limited Liability Company (hereinafter "Purchaser").

Whereas Seller is the owner of full right and title (both legal and equitable) to certain inventions, patents, and applications, defined herein as "Seller Patents";

Whereas, Seller is a party to an action, styled Network Commerce, Inc. v. Microsoft Corporation, Civil Action No. C01-1991P, in the District Court for the Western District of Washington ("the Microsoft Case"); and,

Whereas Purchaser is desirous of acquiring the entire domestic and foreign right title and interest in and to such Seller Patents and all rights of Seller in and to its claims and causes of action in the Microsoft Case.

Now, therefore, Seller and Purchaser hereby covenant and agree as follows:

1. Definitions

1.1. "Seller Patents" shall mean the patents and applications identified on Exhibit A, including (i) all U.S. and foreign patents and patent applications that claim priority to such patents and all U.S. and foreign patents and applications to which such patents and applications relate or claim priority, and (ii) any continuations, continuations-in-part, divisions, reissue applications, extensions, patent Cooperation Treaty applications, or derivatives of any of the foregoing, both foreign and domestic.

1.2. "Prosecution History Files" shall mean all files, documents and tangible things, as those terms have been interpreted pursuant to Federal Rule of Civil Procedure 34, constituting, comprising or relating to investigation, evaluation, prosecution, filing and registration of the Patents, and specifically includes e-mail messages and other electronic or computer stored or generated data.

1.3. "Microsoft Claim" shall mean the claims and causes of action of Seller against Microsoft Corporation which were asserted, or which could have been asserted, by Seller in the Microsoft Case, including (i) all rights to any recoveries of damages, costs or other monetary or other relief, (ii) the right to maintain and prosecute any appeal from any order in the name of the Purchaser or at Purchaser's election in the name of Seller and (iii) the right to maintain and prosecute any further proceedings against Microsoft

EXHIBIT A

in the District Court of the Microsoft Case or in any other court, in the name of the Purchaser or at Purchaser's election in the name of Seller.

1.4. "Net Receipts" shall mean the total recoveries actually received by Purchaser from the exploitation of the Purchased Assets (as defined below in Section 2.3), including all recoveries from licensing, sales, enforcement, lawsuits or settlements, including without limitation, recoveries in the form of cash, stock, personal property, credits or benefits; less all costs and expenses incurred with third parties and not as in-house overhead in connection with prosecuting, licensing, enforcing or defending the Purchased Assets, including without limitation (A) attorneys' and paralegal fees (whether on an hourly or contingent basis and whether for general or local counsel), costs and disbursements, (B) the fees and costs of consultants, experts or technical advisors (other than principals of Purchaser or its affiliates), (C) travel and lodging expenses, (D) duplicating, secretarial, stenographer, postage, courier and similar expenses, (E) filing fees and other Patent Office fees or costs, (F) court costs, (G) legal and other costs related to any re-examination or reissue proceeding or in prosecuting any foreign application, (H) legal and other costs incurred in defending any action or counterclaim in respect of the Patents or the Microsoft Claim, (I) legal and other costs in prosecuting or processing any application, continuing application or continuation in part and (J) patent maintenance fees.

2. Transfer of Rights

2.1. For good and valuable consideration, the receipt and sufficiency of which are hereby acknowledged, Seller hereby agrees to assign and transfer to Purchaser and its representatives, successors and assigns its full and exclusive right, title and interest in and to (i) all Seller Patents, (ii) the Microsoft Claim and (iii) the Microsoft Case. Seller also hereby agrees to assign and transfer to Purchaser and its representatives, successors and assigns its full and exclusive right, title and interest in and to all protectable (e.g., as by patenting) inventions, in the U.S. and every foreign country, described or embodied in the Seller Patents.

2.2. Seller hereby agrees to assign and transfer to Purchaser and its representatives, successors and assigns the full and exclusive right to recover all past damages and other potential relief arising from (i) infringement of the Seller Patents, (ii) the Microsoft Claim and (iii) the Microsoft Case.

2.3. The closing (the "Closing") of the assignment and transfer of the Seller Patents, the Microsoft Claim and other assets described in Section 2.1 and 2.2 (the "Purchased Assets") shall take place on the second business day

following the satisfaction of the conditions set forth in Section 2.4 through 2.6 and Section 4.9 hereof.

2.4. For the purpose of recordation and in accordance with the transfers herein, at the Closing, Seller shall execute separate assignment documents listing the Seller Patents. Upon the written request of the Purchaser and without additional charge or at the Purchaser's expense, the Seller shall execute and deliver to the Purchaser all such additional instruments of transfer, conveyance, endorsement and assignment (in a form satisfactory to the Purchaser) as shall be necessary to transfer the Patents to Purchaser.

2.5. Effective upon the Closing, Seller conveys to Purchaser, its representatives, successors and assigns, the right to make applications on their own behalf for protection of the inventions conveyed herein in the U.S. and foreign countries and to claim, under United States law, the Patent Cooperation Treaty, the International Convention and/or other international arrangements for any such application, priority to any earlier application or patent.

2.6. Within 30 days of the Closing, Seller shall transfer to Purchaser all Prosecution History Files and related files maintained by Seller and its outside and in-house counsel.

3. Payment

3.1. As consideration for the assignment of the Seller Patents and other rights granted by Seller herein, Purchaser shall (i) pay to Seller on or prior to the Closing, the total sum of One Hundred Twenty Thousand U.S. Dollars ($120,000.00) and (ii) grant Seller, its successors and assigns a perpetual right to receive an amount equal to 5% of the Net Receipts (together herein referred to as the "Purchase Price"). Upon the request of Seller, its bankruptcy representative, or its designee, Purchaser and its representatives, successors, and assigns shall provide Seller or its designee with appropriate reports detailing recoveries, costs, and expenses related to Net Receipts, and Seller, its bankruptcy representative, or its designee shall have the right, at its expense, to audit the records pertaining to such recoveries, costs, and expenses.

3.2. Payments under Paragraph 3.1 shall be made by electronic funds transfer. Such payment shall be deemed to be made on the date credited to the following account or to such other account of which Seller may notify Purchaser in writing:

Bank:	Washington Trust Bank
Address:	601 Union Street, Suite 4747

	Seattle, WA 98101
Account Name:	Network Commerce Inc.
ABA Routing #:	125100089
Account #:	2306914618
Bank Contact:	Kirsten Imori

4. Covenants and other Provisions

4.1. Seller represents and warrants that (a) it has the right to assign the Purchased Assets, (b) it is conveying through this Agreement its undivided right, title and interest in and to the Purchased Assets and that no other party has any claim of ownership to the Purchased Assets or any security interest, encumbrance, lien or other claim in or to the Purchased Assets, and (c) no licenses, sublicenses, covenants not to sue or other rights have been granted with respect to the Seller Patents, and no entity has licenses or rights under 11 U.S.C. Section 365(n), except for the licenses and other rights contained in the agreements identified on Exhibit B.

4.2. Seller represents and warrants that no agreements with third parties prevent Seller from entering into this Agreement.

4.3. Seller represents and warrants that, to its knowledge, it has not taken, and will not take, any action materially adversely affecting the validity, enforceability, or issuance of the Seller Patents.

4.4. Seller represents and warrants that none of the Seller Patents is involved in any interference or opposition proceeding and, to Seller's knowledge, no such proceeding is being threatened with respect to any such Seller Patents.

4.5. Seller represents and warrants that subject to appropriate order of the Bankruptcy Court, it is able to convey the Purchased Assets free and clear of any liens, encumbrances, security interests, or other claims (including any claims by Seller's current or former attorneys for fees or costs relating to any matter including the Microsoft Case) to the fullest extent of the Bankruptcy Court's authority to so order, except for the licenses noted on Schedule B.

4.6. Seller shall pay all transfer taxes imposed on the sale of the Purchased Assets, including all sales, gross receipts, excise and gross income taxes.

4.7. Subject to the authority and jurisdiction of the Bankruptcy Court and except as is consistent with the applicable orders of the Bankruptcy Court with respect to the procedures relating to the sale of its assets, Seller covenants and agrees that it shall not execute any writing or do any act whatsoever conflicting with the terms of this Agreement, and that,

following the Closing, Seller will at any time upon request, without further or additional consideration, but at the expense of Purchaser, execute such additional assignments or other writings and perform such additional acts as Purchaser may deem reasonably necessary to perfect Purchaser's ownership of the Purchased Assets. Seller further covenants and agrees, at Purchaser's expense, to render all reasonably necessary assistance following the Closing in making application for, prosecuting in any patent office internationally, and obtaining original, continuation, continuation-in-part, divisional, reissued, reexamined, and national phase patents of the U.S. or of any and all foreign countries on the inventions assigned herein, and in enforcing any rights or choses in action accruing as a result of the rights assigned herein, including enforcing claims in the Microsoft Case and the appeal of the District Court's summary judgment, and by executing statements and other affidavits, it being understood that the foregoing covenant and agreement shall bind, and inure to the benefit of, the assigns and representatives of all parties hereto.

4.8. This Agreement and all matters relating to this Agreement shall be construed and controlled by the laws of the State of Washington. If any legal proceeding or other legal action relating to the Agreement is brought or otherwise initiated, the venue therefore will be the Bankruptcy Court. Purchaser and Seller hereby expressly and irrevocably consent and submit to the jurisdiction of the Bankruptcy Court.

4.9. The Closing and the transactions contemplated herein are and shall be contingent upon (i) the issuance by the Bankruptcy Court of an order, in a form reasonably satisfactory to Purchaser, approving the transactions provided for herein free and clear of liens, encumbrances and rights, to the fullest extent of the Bankruptcy Court's authority to so order (the "Sale Order"); (ii) execution and delivery of the documents and other instruments required to be delivered by Purchaser and Seller on or prior to Closing pursuant to this Agreement; and (iii) receipt by Seller of $120,000.00 (the portion of the Purchase Price to be paid at Closing). The Sale Order shall contain, among other things, a finding that the sale of the Purchased Assets to Purchaser is in good faith within the meaning of Bankruptcy Code Section 363(m).

4.10. Except as otherwise provided in the Agreement, the parties shall pay their respective expenses incurred in connection with the preparation, execution, and delivery of this Agreement and the consummation of the transactions contemplated hereby.

4.11. All notices, requests, demands, and other communications hereunder shall be deemed to have been duly given on the day they are (i) deposited in the U.S. mail, postage prepaid, certified or registered, return receipt requested;

or (ii) sent by air express courier, charges prepaid, and addressed as follows:

4.11.1. If to Purchaser: CRS LLC: Attention Reed Corry, 600 University Street, Suite 2800, Seattle WA 98101

4.11.2. If to Seller: Network Commerce Inc: John R. Knapp, Jr., Cairncross & Hempelmann, P.S., 524 Second Avenue, Suite 500, Seattle, WA 98104-2323.

4.11.3. Such addresses may be changed, from time to time, by means of a written notice delivered by the party seeking to change such address in the manner provided for in this paragraph.

4.12. This Agreement shall be binding upon and inure to the benefit of the parties and their respective successors and assigns.

4.13. This Agreement constitutes the entire agreement between the parties and supersedes all prior agreements and understandings, oral and written, among the undersigned with respect to the subject matter hereof.

In witness whereof, the parties hereto have caused this assignment to be made and executed by duly authorized officers as of the dates indicated below.

Agreed to: **Network Commerce Inc.**	*Agreed to:* **CRS, LLC**
By:_____	By:_____
Name:_____	Name: Reed Corry
Title:_____	Title: Member
Date:_____	Date:_____

or (ii) sent by air express courier, charges prepaid, and addressed as follows:

4.11.1. If to Purchaser: CRS LLC: Attention Reed Corry, 600 University Street, Suite 2800, Seattle WA 98101

4.11.2. If to Seller: Network Commerce Inc: John R. Knapp, Jr., Cairncross & Hempelmann, P.S., 524 Second Avenue, Suite 500, Seattle, WA 98104-2323.

4.11.3. Such addresses may be changed, from time to time, by means of a written notice delivered by the party seeking to change such address in the manner provided for in this paragraph.

4.12. This Agreement shall be binding upon and inure to the benefit of the parties and their respective successors and assigns.

4.13. This Agreement constitutes the entire agreement between the parties and supersedes all prior agreements and understandings, oral and written, among the undersigned with respect to the subject matter hereof.

In witness whereof, the parties hereto have caused this assignment to be made and executed by duly authorized officers as of the dates indicated below.

Agreed to:	*Agreed to:*
Network Commerce Inc.	**CRS, LLC**
By: [signature]	By: _____
Name: N Scott Dickson	Name: Reed Corry
Title: CFO	Title: Member
Date: 11/24/03	Date: _____

or (ii) sent by air express courier, charges prepaid, and addressed as follows:

4.11.1. If to Purchaser: CRS LLC: Attention Reed Corry, 600 University Street, Suite 2800, Seattle WA 98101

4.11.2. If to Seller: Network Commerce Inc: John R. Knapp, Jr., Calfacross & Hempelmann, P.S., 524 Second Avenue, Suite 500, Seattle, WA. 98104-2323.

4.11.3. Such addresses may be changed, from time to time, by means of a written notice delivered by the party seeking to change such address in the manner provided for in this paragraph.

4.12. This Agreement shall be binding upon and inure to the benefit of the parties and their respective successors and assigns.

4.13. This Agreement constitutes the entire agreement between the parties and supersedes all prior agreements and understandings, oral and written, among the undersigned with respect to the subject matter hereof.

In witness whereof, the parties hereto have caused this assignment to be made and executed by duly authorized officers as of the dates indicated below.

Agreed to:
Network Commerce Inc.

By:_____

Name:_____

Title:_____

Date:_____

Agreed to:
CRS, LLC

By:_____

Name: Reed Corry

Title: Member

Date: 11/24/2003

The Honorable Philip H. Brandt
Chapter 11
Response Date: December 5, 2003

UNITED STATES BANKRUPTCY COURT FOR THE
WESTERN DISTRICT OF WASHINGTON AT SEATTLE

In re: NETWORK COMMERCE INC., Debtor.	NO. 02-23396-PHB11 ORDER APPROVING SALE UNDER ORDER AUTHORIZING DEBTOR TO CONDUCT SALE OF PATENTS, MICROSOFT CLAIM, AND MICROSOFT CASE PURSUANT TO AUCTION AND SECTION 363(b), (f), AND (m) OF THE BANKRUPTCY CODE

THIS MATTER having come before the Court on the Debtor's Motion for Order Authorizing Debtor to Sell Patents, Microsoft Claim, and Microsoft Case to TechSearch, LLC Pursuant to Section 363(b), (f), and (m) of the Bankruptcy Code (the "Motion"); the Court having entered the Order Authorizing Debtor to Conduct Sale of Patents, Microsoft Claim, and Microsoft Case Pursuant to Auction and Section 363(b), (f), and (m) of the Bankruptcy Code on November 20, 2003 (the "Auction Order"), the definitions, terms, and conditions of which are incorporated herein and made applicable hereto by this reference; the Debtor having conducted

Cairncross & Hempelmann, P.S.
Law Offices
524 Second Avenue, Suite 300
Seattle, Washington 98104-2323
Phone: 206-587-0700 • Fax: 206-587-2308

EXHIBIT B

the Auction pursuant to the terms of the Auction Order; the Debtor having filed a notice identifying the Successful Bidder as CRS, LLC, with a copy of the Agreement attached, pursuant to paragraph 2 of the Auction Order; the Court finding that the sale of the Assets is based on sound business justifications, and such sale is in the best interests of the Debtor's estate based on the results of the Auction and for the reasons set forth in the Motion, the Memorandum, and the Dickson Declaration; the Court further finding that the sale of the Assets pursuant to the Agreement has been proposed and, if consummated, will have been consummated in good faith in accordance with 11 U.S.C. § 363(m); the Court further finding that the Successful Bidder is a good faith purchaser and is entitled to the protections afforded under 11 U.S.C. § 363(m) and that if the transaction contemplated by the Agreement closes, the Successful Bidder will have acted in good faith in closing the transaction; the Court further finding that the Successful Bidder is not an insider or affiliate of the Debtor; the Court further finding that the consideration to be received by the Debtor from the Successful Bidder is fair and reasonable, and that the sale does not unfairly benefit insiders, a proprietary purchaser, or any creditor or class of creditors; and the Court further finding that consummation of the Agreement is in the best interests of the Debtor, its estate, all creditors, and other parties in interest,

NOW, THEREFORE, it is HEREBY ORDERED:

1. The Agreement is approved, and the Debtor is authorized to enter into the Agreement. A copy of the Agreement is attached hereto as Exhibit 1.

2. Except as otherwise set forth by this Order or the Agreement, in accordance with 11 U.S.C. § 363(f), the sale of the Assets pursuant to the Agreement is and shall be free and clear of any interest in such Assets of an entity other than the estate (whose ongoing interest with respect to the Assets shall be limited to the extent of the Purchase Price as expressly provided in

Cairncross & Hempelmann, P.S.
Law Offices
524 Second Avenue, Suite 500
Seattle, Washington 98104-2323
Phone: 206-587-0700 • Fax: 206-587-2308

the Agreement), including but not limited to all claims, liens, and encumbrances of any nature, kind, or description, with such interests, claims, liens, and encumbrances to attach to the sale proceeds in the order and priority that existed prior to the sale.

3. The Agreement was proposed, negotiated, and entered into in good faith after arm's length bargaining by the parties and the Auction and provides the Debtor with the highest or otherwise best offer received for the Assets. The Successful Bidder is a good faith purchaser pursuant to 11 U.S.C. § 363(m) and entitled to the protections thereunder.

4. The Debtor is authorized to take such further actions as may be necessary to implement, close, and consummate the sale of the Assets pursuant to the terms of the Agreement without further notice or order of this Court.

5. The Successful Bidder has not assumed or otherwise become obligated for any of the Debtor's liabilities. All creditors of the Debtor, whether known or unknown, are hereby enjoined from asserting or prosecuting a claim or cause of action against the Successful Bidder or the Assets to recover on account of any liability owed by the Debtor.

6. This Order shall not be subject to the stay or Federal Rule of Bankruptcy Procedure 6004(g).

7. The Successful Bidder shall pay the Purchase Price (as defined in the Agreement) pursuant to the terms of the Agreement.

8. The Successful Bidder shall not be deemed a successor to the Debtor, by reason of the Closing (as defined in the Agreement) or otherwise.

///

Cairncross & Hempelmann, P.S.
Law Offices
524 Second Avenue, Suite 500
Seattle, Washington 98104-2323
Phone: 206-587-0700 • Fax: 206-587-2308

9. Federal, state, and local governmental units are directed to accept any and all documents to effectuate the Closing.

DATED this 9 day of December, 2003.

The Honorable Philip H. Brandt
United States Bankruptcy Judge

Presented by:

Cairncross & Hempelmann, P.S.

/s/ John R. Rizzardi
John R. Rizzardi, WSBA # 9388

Attorneys for Debtor Network Commerce Inc.

Cairncross & Hempelmann, P.S.
Law Offices
524 Second Avenue, Suite 500
Seattle, Washington 98104-2323
Phone: 206-587-0700 • Fax: 206-587-2308

ASSIGNMENT

WHEREAS, Network Commerce Inc. is the assignee and holds the entire right, title and interest to and in certain new and useful inventions which have resulted in the issuance of certain United States Patent Numbers as set forth in Exhibit A, and for which certain additional related continuation, divisional, continuation-in-part and reissue applications have been filed; as set forth in Exhibit A;

NOW THEREFORE, Network Commerce, Inc., for and in consideration of certain good and valuable consideration, the sufficiency and receipt of which is hereby acknowledged, at the request of the assignee does sell, assign and transfer unto said assignee, CRS LLC a Washington Limited Liability Company, having a place of business at 2800 One Union square, 600 University Street, Seattle, WA 98101, its successors, legal representatives and assigns, the aforesaid patents and all continuation, divisional, continuation-in-part and reissue applications, all patent applications in foreign countries, all applications pursuant to the Patent Cooperation Treaty, and all applications for extension filed or to be filed for the invention, and all Letters Patent, Invention Registrations, Utility Models, Extensions or Reissues and other patent rights, obtained for the inventions in the United States or any other country; it also assigns any right, title or interest in and to the inventions which have not already been transferred to the assignee; it warrants that it has made no assignment of the inventions, applications or patents therefor to a party other than CRS LLC and it is under no obligation to make any assignment of the invention, application, or patent therefor to any other party; and it further agrees to cooperate with the assignee hereunder in the obtaining and sustaining of any and all such Letters Patent and in confirming assignee's exclusive ownership of the inventions, but at the expense of said assignee.

The Commissioner of Patents is hereby authorized and requested to issue the Letters Patent solely in accordance with the terms of this Assignment, to CRS LLC its successors, legal representatives and assigns, as the assignee of the entire right, title and interest therein.

IN WITNESS WHEREOF, the parties hereto have executed this Assignment as of the date indicated hereunder.

Date: 1/21/04

Network Commerce, Inc,

By: _____

Its: Chief Financial Officer

EXHIBIT C

STATE OF)
) ss.
COUNTY OF)

Before me, a Notary Public in and for said County and State, personally appeared _____, known to me to be the person whose name is subscribed to the foregoing instrument, and acknowledged to me that he executed the same for the purposes and considerations therein expressed.

Given under my hand and seal of office this _____ day of _____, 2004.

_____
Notary Public

My Commission expires: _____

Exhibit A

Patent 6,073,124
Method and System for Securely Incorporating Electronic Information into an Online Purchasing Application
Date Issued: June 6, 2000

Patent 6,141,698
Method and System for Injecting New Code into Existing Application Code
Date Issued: October 31, 2000.

Patent 6,266,681
Method and System for Injecting Code to Conditionally Incorporate A User Interface Component In An HTML Document
Date Issued: April 8, 1999

Patent 6,401,077 B1
Method and System for Providing Additional Behavior Through a Web Page
Date Issued: June 4, 2002

Patent 6,405,316
This is Sub of the '698 patent – Method and System for Injecting New Code into Existing Application Code
Date Issued: June 28, 2000

U.S. Patent Application No. 10/406,624
Date Filed: April 2, 2003

U.S. Patent Application No. 10/133,213
Method and System for Injecting New Code into Existing Application Code
Date Filed: April 26, 2002.

U.S. Patent Application No. 09/872,474
Method and System for Injecting Code to Conditionally Incorporate a User Interface Component in an HTML Document
Date Filed: June 1, 2001

U.S. Patent Application No. 10/091,341
Method and System for Providing Additional Behavior Through a Web Page
Date Filed: March 4, 2002

U.S. Patent Application No. 09/464,662
Method and System for Securely Incorporating Electronic Information into an Online Purchasing Application

Date Filed: December 15, 1999

Exhibit A

continued...

U.S. Patent Application No. 09/543,105
Maintaining Wish Lists Across Multiple Vendors via a Scraper Engine
Date Filed: April 5, 2000

U.S. Patent Application No. 09/543,219
Providing Unique Web Site Capabilities
Date Filed: April 5, 2000

U.S. Patent Application No. 09/543,221
Scraper Template Creation and Maintenance
Date Filed: April 5, 2000

ASSIGNMENT

Pursuant to the Asset Purchase Agreement dated November 24, 2003, Network Commerce, Inc., a Washington corporation, for valuable consideration hereby assigns all of its right, title and interest in any and all claims and causes of action which were or could have been asserted by Network Commerce Inc. against Microsoft Corporation in *Network Commerce Inc. v. Microsoft Corporation*, No. C01-1991P including i) all rights to any recoveries of damages, costs or other monetary or other relief, ii) the right to maintain and prosecute any appeal from any order in the name of CRS LLC or at CRS's election in the name of Network Commerce Inc. and iii) the right to maintain and prosecute any further proceedings against Microsoft in the name of CRS LLC or at CRS LLC's election, in the name of Network Commerce Inc. Network Commerce, Inc. agrees to execute and have executed this and any other document necessary to transfer its above-referenced claims to CRS LLC, and to perfect CRS LLC's ownership of them.

DATED this 21 day of January, 2004

Network Commerce, Inc.

By: _____

Its: Chief Financial Officer

THE HONORABLE MARSHA J. PECHMAN

UNITED STATES DISTRICT COURT
WESTERN DISTRICT OF WASHINGTON
AT SEATTLE

NETWORK COMMERCE, INC., a Washington corporation; and CRS, LLC, a Washington limited liability company, Plaintiffs, v. MICROSOFT CORPORATION, a Washington corporation, Defendant.	NO. C01-1991P COMPLAINT FOR PATENT INFRINGEMENT JURY DEMAND

Plaintiffs Network Commerce, Inc. ("Network Commerce") and CRS, LLC ("CRS"), in and for their complaint against defendant Microsoft Corporation ("Microsoft"), allege as follows:

THE PARTIES

1. Plaintiff Network Commerce is a Washington corporation with its principal place of business in Seattle, Washington.

2. Plaintiff CRS is a Washington limited liability company with its principle place of business in Seattle, Washington.

3. Defendant Microsoft is a Washington corporation with its principal place of business at One Microsoft Way in Redmond, Washington. Microsoft conducts business and,

EXHIBIT D

ROHDE & VAN KAMPEN PLLC
1000 Second Avenue, Suite 3110
Seattle, Washington 98104-1046
(206) 386-7353

upon information and belief, has committed or actively induced the allegedly infringing acts described below in this judicial district.

JURISDICTION AND VENUE

4. This is an action for patent infringement under the patent laws of the United States, 35 U.S.C. §271 et seq. Jurisdiction is conferred upon this Court by 28 U.S.C. §§1331 (federal question), 1332 (diversity) and 1338(a)(patent case).

5. Venue is proper in this District under 28 U.S.C. §1391(c) and 1391(d) because defendant is subject to jurisdiction in this district for the claims alleged herein.

PLAINTIFFS AND THEIR RIGHTS

6. Network Commerce is a leading provider of technology infrastructure services for businesses, merchants, Internet sites and wireless networks conducting commerce over the Internet. Network Commerce also licenses its technology.

7. On June 6, 2000, United States Patent No. 6,073,124 (the "'124 patent") was duly and legally issued to Network Commerce. The '124 patent discloses and claims methods and systems for conducting electronic commerce, including a method that uses separate servers and a download component to coordinate the download of information for on-line transactions.

8. On November 1, 2002, Network Commerce filed for protection of the bankruptcy courts.

9. In November 2003, CRS agreed to purchase the patents of Network Commerce, including the '124 patent, and Network Commerce's patent infringement claim against Microsoft. The agreement was approved by the Bankruptcy Court on December 9, 2003. On January 21, 2004, the '124 patent was assigned to CRS.

10. The '124 patent is valid and enforceable.

ROHDE & VAN KAMPEN PLLC
1000 Second Avenue, Suite 3110
Seattle, Washington 98104-1046
(206) 386-7353

DEFENDANT'S INFRINGEMENT

11. Microsoft is in the business of, among other things, developing software products, including, but not limited to, software products that deliver digital media over the Internet. On information and belief, Microsoft uses in certain products, among other methods, a method by which one web server downloads a file or files that act to coordinate the downloading of digital media from another web server.

12. On information and belief, Microsoft has offered for sale, distributed, used, and/or sold software, including its Windows Media products, that directly perform or induce the performance of the methods claimed in one or more claims of the '124 patent when correctly construed, including at least claims 1, 7, 11, 13, 14, and 16.

CLAIM FOR PATENT INFRINGEMENT

13. Plaintiffs reallege and incorporate by reference as if fully set forth herein the allegations in paragraphs 1-9 above.

14. Microsoft has infringed one or more claims of the '124 patent.

15. On information and belief, Microsoft's infringement has been deliberate and willful.

16. Microsoft will continue such infringement unless enjoined by this Court.

PRAYER FOR RELIEF

WHEREFORE, Plaintiffs pray for the following relief:

1. For a preliminary and permanent injunction, enjoining Microsoft, its agents, servants, employees and all those in privity with Microsoft from making, using, selling or offering for sale any method or system infringing the '124 patent;

2. For judgment in favor of Plaintiffs and against Microsoft awarding damages together with interest and costs to compensate for the infringement by Microsoft of the '124

ROIDE & VAN KAMPEN PLLC
1000 Second Avenue, Suite 3110
Seattle, Washington 98104-1046
(206) 386-7353

patent, such award to be tripled in accordance with 35 U.S.C. §284 or as otherwise permitted by law;

3. For judgment against Microsoft for Plaintiffs' costs of suit, including Network Commerce's reasonable attorneys' fees pursuant to 35 U.S.C. §285 or as otherwise permitted by law; and

4. For such other relief as the Court may deem just and proper.

JURY DEMAND

Pursuant to Federal Rule of Civil Procedure 38(b), Plaintiffs demand a trial by jury as to all issues so triable in this action.

DATED this ____ day of March, 2004.

ROHDE & VAN KAMPEN, PLLC

By: _____
Robert E. Rohde, WSBA #12809
Attorneys for CRS, LLC

ROHDE & VAN KAMPEN PLLC
1000 Second Avenue, Suite 3110
Seattle, Washington 98104-1046
(206) 386-7353

EXHIBIT E

UNITED STATES DISTRICT COURT
WESTERN DISTRICT OF WASHINGTON
AT SEATTLE

NETWORK COMMERCE, INC,

Plaintiff,

v.

MICROSOFT CORPORATION,

Defendant.

No. C01-1991P

ORDER GRANTING MOTION TO INTERVENE

This matter comes before the Court on CRS, LLC's motion to intervene as additional plaintiff, as CRS has purchased rights to this lawsuit from Network Commerce, Inc. (Dkt. No. 91). There being no opposition to the motion, the Court hereby GRANTS the motion. The Court will make an independent determination on plaintiff NCI's motion for substitution of CRS as plaintiff once the briefing on that motion is complete.

The Clerk is directed to send copies of this order to all counsel of record.

Dated: April 21, 2004

/s/ Marsha J. Pechman
Marsha J. Pechman
United States District Judge

EXHIBIT F

THE HONORABLE RICHARD A. JONES

UNITED STATES DISTRICT COURT
WESTERN DISTRICT OF WASHINGTON
AT SEATTLE

CRS, LLC, Plaintiff, v. VALVE CORPORATION, Defendant.	No. C08-0361RAJ **DEFENDANT VALVE CORPORATION'S ANSWER AND COUNTERCLAIM** **JURY DEMAND**

Defendant Valve Corporation ("Valve") hereby answers the numbered paragraphs of Plaintiff CRS, LLC's ("CRS") Complaint for Patent Infringement as follows:

I. JURISDICTION AND VENUE

1. Valve admits only that this lawsuit purports to be an action for patent infringement. Valve denies the remaining allegations in paragraph 1.

2. Valve admits jurisdiction as to 28 U.S.C. §§ 1331 and 1338(a). Valve denies jurisdiction as to 28 U.S.C. § 1332.

3. Valve admits the allegations in paragraph 3.

Riddell Williams P.S.
1001 FOURTH AVENUE
SUITE 4500
SEATTLE, WA 98154-1192
206.624.3600

II. THE PARTIES

4. Valve lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 4 and, therefore, denies the same.

5. Valve admits the allegations in paragraph 5.

6. Valve admits that on its face the title page of the '124 Patent states that it was issued on June 6, 2000. Valve lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 6 and, therefore, denies the same.

III. PATENT INFRINGEMENT

7. Valve hereby incorporates the prior paragraphs of its answer as if fully set forth herein.

8. Valve admits that it maintains websites, including a website at www.steampowered.com, that are accessible over the internet. Valve admits that www.steampowered.com, other websites and computer systems are used to purchase games and other items over the internet. Valve denies the remaining allegations in paragraph 8.

9. Valve denies the allegations in paragraph 9.

10. Valve denies the allegations in paragraph 10.

11. Valve denies the allegations in paragraph 11.

VALVE'S AFFIRMATIVE AND OTHER DEFENSES

Valve asserts the following defenses. Valve reserves the right to amend its answer with additional defenses as further information is obtained.

First Defense

The Complaint, in whole or in part, fails to state a claim upon which relief can be granted.

Riddell Williams P.S.
1001 FOURTH AVENUE
SUITE 4500
SEATTLE, WA 98154-1192
206.624.3600

Second Defense

Valve has not infringed, contributed to the infringement of, or induced the infringement of U.S. Patent No. 6,073,124 ("the '124 Patent") and is not liable for infringement thereof. Without admitting any infringement, Valve states that any and all Valve products or actions that are accused of infringement have substantial uses that do not infringe and therefore cannot induce or contribute to the infringement of the '124 Patent.

Third Defense

On information and belief, the '124 Patent is invalid and/or unenforceable for failing to comply with the provisions of the Patent Laws contained in Title 35 U.S.C., including without limitation one or more of 35 U.S.C. §§ 102, 103, and 112.

Fourth Defense

Plaintiff's alleged cause of action for patent infringement is barred under the doctrine of prosecution history estoppel, and Plaintiff is estopped from claiming that the '124 Patent covers or includes any accused Valve product or method.

Fifth Defense

On information and belief, Plaintiff has failed to plead and meet the requirements of 35 U.S.C. § 271(b) and (c), and is not entitled to any alleged damages prior to providing any actual notice to Valve of the '124 Patent.

Sixth Defense

On information and belief, Plaintiff has failed to plead and meet the requirements of 35 U.S.C. § 284 for enhanced damages and is not entitled to any damages prior to providing any actual notice to Valve of the '124 Patent and any alleged infringement thereof.

Seventh Defense

Plaintiff's claim is barred, in whole or in part, by the equitable doctrines of laches, unclean hands, estoppel, ratification, and/or waiver.

Riddell Williams P.S.
1001 FOURTH AVENUE
SUITE 4500
SEATTLE, WA 98154-1192
206.624.3600

Eighth Defense

On information and belief, Plaintiff has failed to plead and meet the requirements of 35 U.S.C. §287, and has otherwise failed to show it is entitled to any damages.

COUNTERCLAIMS

COUNT I

DECLARATORY JUDGMENT OF NONINFRINGEMENT

1. This action arises under the Patent Laws of the United States, Title 35 U.S.C. §§ 1, et seq. This Court has subject matter jurisdiction over this counterclaim under 28 U.S.C. §§ 1338, 2201, and 2202. Venue is proper in this District under 28 U.S.C. § 1391(b).

2. Valve Corporation ("Valve") is a Washington corporation with its principal place of business in Bellevue, Washington.

3. CRS purports to be a Washington Limited Liability Company with its principal place of business in Seattle, Washington.

4. CRS purports to own the '124 Patent.

5. CRS alleges that Valve has infringed the '124 Patent.

6. An actual controversy, within the meaning of 28 U.S.C. §§ 2201 and 2202, exists between Valve, on the one hand, and CRS, on the other, with respect to the infringement or noninfringement of the '124 Patent.

7. Valve has not infringed, or induced the infringement of, the '124 Patent, and is not liable for infringement thereof.

COUNT II

DECLARATORY JUDGMENT OF INVALIDITY OF THE '124 PATENT

8. Valve incorporates by reference its responses as set forth above in the answers to paragraphs 1 through 7 of its Counterclaims.

Riddell Williams P.S.
1001 FOURTH AVENUE
SUITE 4500
SEATTLE, WA 98154-1192
206.624.3600

9. The '124 Patent, and each claim thereof, is invalid for failing to comply with the provisions of the Patent Laws, including one or more of 35 U.S.C. §§ 102, 103, and 112.

10. An actual controversy, within the meaning of 28 U.S.C. §§ 2201 and 2202, exists between Valve, on the one hand, and CRS, on the other, with respect to whether the claims of the '124 Patent are invalid.

JURY DEMAND

Valve demands a trial by jury.

DEMAND FOR RELIEF

WHEREFORE, Defendant Valve Corporation, having answered Plaintiff CRS, LLC's Complaint, requests this Court to:

(1) Enter judgment against CRS, and dismiss CRS's Complaint with prejudice;

(2) Enter judgment declaring that Valve has not infringed, contributed to the infringement of, or induced infringement of the '124 Patent;

(3) Enter judgment declaring that the '124 Patent is invalid;

(4) Award Valve prejudgment interest and the costs of this action;

(5) Award Valve its cost of suit and attorneys' fees to the fullest extent allowed by law; and

(6) Grant such other relief as the Court deems just and appropriate.

//

//

//

//

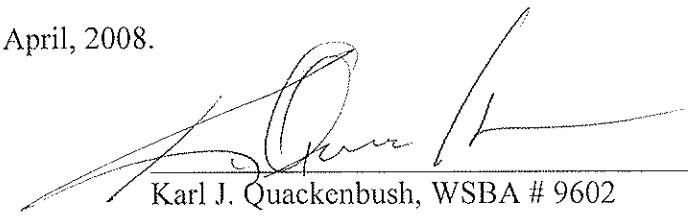
//

//

//

Riddell Williams P.S.
1001 FOURTH AVENUE
SUITE 4500
SEATTLE, WA 98154-1192
206.624.3600

DATED this 24th day of April, 2008.



Karl J. Quackenbush, WSBA # 9602
Jayson W. Sowers, WSBA # 27618
RIDDELL WILLIAMS P.S.
1001 Fourth Avenue, Suite 4500
Seattle, WA 98154
Tel: (206) 624-3600
Fax: (206) 389-1708
kquackenbush@riddellwilliams.com
jsowers@riddellwilliams.com

J. Christopher Carraway, WSBA # 37944
Klarquist Sparkman, LLP
One World Trade Center
121 S.W. Salmon Street, Suite 1600
Portland, OR 97204
Tel: (503) 226-7391
Fax: (503) 228-9446
christopher.carraway@klarquist.com

Inge Larish, WSBA # 34954
Klarquist Sparkman, LLP
One Union Square
600 University Street, Suite 2950
Seattle, WA 98101
Tel: (206) 264-2960
Fax: (206) 624-2719
inge.larish@klarquist.com

Attorneys for Defendant
VALVE CORPORATION

CERTIFICATE OF SERVICE

HOLLY ROHR TRAN states as follows:

I am over 18 years of age and a citizen of the United States. I am employed as a legal secretary by the law firm of Riddell Williams P.S.

On the date noted below I electronically filed the foregoing document titled DEFENDANT VALVE CORPORATION'S ANSWER AND COUNTERCLAIM using the CM/ECF system and caused to be delivered true and accurate copies of the same upon the following:

Karl Justin Quackenbush
kquackenbush@riddellwilliams.com,lwerner@riddellwilliams.com

Robert E Rohde
brohde@rohdelaw.com,npaine@rohdelaw.com,jrogers@rohdelaw.com

Jayson W Sowers
jsowers@riddellwilliams.com,mdowns@riddellwilliams.com,
hrohrtran@riddellwilliams.com

I declare under penalty of perjury under the laws of the State of Washington that the foregoing is true and correct and that this Certificate of Service was executed on April 24, 2008, at Seattle, Washington.

Holly Rohr Tran

Riddell Williams P.S.
1001 FOURTH AVENUE
SUITE 4500
SEATTLE, WA 98154-1192
206.624.3600

EXHIBIT G

US006073124A

United States Patent [19]

Krishnan et al.

[11] Patent Number: 6,073,124

[45] Date of Patent: Jun. 6, 2000

[54] **METHOD AND SYSTEM FOR SECURELY INCORPORATING ELECTRONIC INFORMATION INTO AN ONLINE PURCHASING APPLICATION**

[75] Inventors: **Ganapathy Krishnan**, Bellevue; **John Guthrie**; **Scott Oyler**, both of Seattle, all of Wash.

[73] Assignee: **ShopNow.com Inc.**, Seattle, Wash.

[21] Appl. No.: **08/895,221**

[22] Filed: **Jul. 15, 1997**

Related U.S. Application Data

[63] Continuation-in-part of application No. 08/792,719, Jan. 29, 1997

[60] Provisional application No. 60/049,844, Jun. 17, 1997.

[51] **Int. Cl.**[7] **G06F 17/60**

[52] **U.S. Cl.** **705/59**; 705/51; 705/26

[58] **Field of Search** 705/1, 18, 21, 705/26, 51, 59; 380/3, 4, 23, 24, 25, 277

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,005,122	4/1991	Griffin et al.	709/203
5,337,357	8/1994	Chou et al.	380/4
5,390,297	2/1995	Barber et al.	364/280
5,530,752	6/1996	Rubin	380/4
5,553,143	9/1996	Ross et al.	380/25
5,592,549	1/1997	Nagel et al.	380/4
5,708,709	1/1998	Rose	380/4
5,710,887	1/1998	Chelliah et al.	705/26

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0 667 572 A1	8/1995	European Pat. Off.	G06F 9/445
0 704 785 A2	4/1996	European Pat. Off.	G06F 1/00
WO 97/14087	4/1997	European Pat. Off. .	
0 778 512 A2	6/1997	European Pat. Off.	G06F 1/00
0 795 809 A2	9/1997	European Pat. Off.	G06F 1/00

OTHER PUBLICATIONS

T. Berners–Lee et al., "Hypertext Transfer Protocol—HTTP 1.0," Request for Comments (RFC) 1945, MIT/LCS, May, 1996.

T. Berners–Lee et al., "Uniform Resource Locators (URL)," RFC 1738, CERN, Xerox PARC, Univ. of Minn., Dec., 1994.

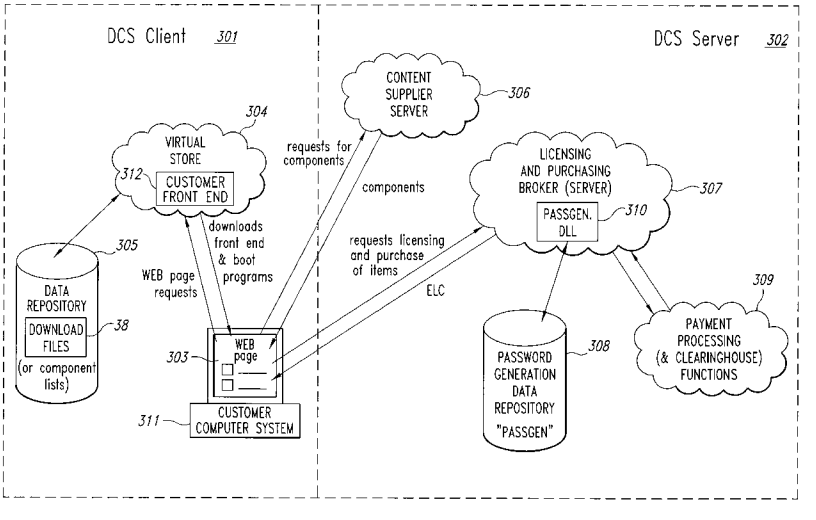
(List continued on next page.)

Primary Examiner—James P. Trammell
Assistant Examiner—Nicholas David Rosen
Attorney, Agent, or Firm—Perkins Coie LLP

[57] **ABSTRACT**

A method and system for facilitating digital commerce using a secure digital commerce system is provided. The secure digital commerce system is arranged according to a client/server architecture and includes a modularized DCS client and DCS server. The DCS client and the DCS server are incorporated into an online purchasing system, such as a virtual store, to perform the purchase and online delivery of electronic content. The DCS client includes a set of components which include a secured copy of the merchandise and various components needed to license and purchase the merchandise and to unsecure and process (e.g., execute) the licensed merchandise. The DCS client communicates with the DCS server to download the components onto a customer's computer system and to license and purchase a requested item of merchandise. The DCS server, which includes a content supplier server, a licensing and purchasing broker, and a payment processing function, supplies merchandise-specific components and licenses the requested item of merchandise by generating an electronic certificate. The electronic certificate contains license parameters that are specific to the requested merchandise and an indicated purchasing option. Once a valid electronic license certificate for the requested merchandise is received by the DCS client, the merchandise is made available to the customer for use in accordance with the licensing parameters contained in the electronic license certificate.

16 Claims, 21 Drawing Sheets



U.S. PATENT DOCUMENTS

5,724,424	3/1998	Gifford	380/24
5,757,908	5/1998	Cooper et al.	380/4
5,758,068	5/1998	Brandt et al.	713/200
5,758,069	5/1998	Olsen	713/201
5,778,173	7/1998	Apte	380/25
5,794,259	8/1998	Kikinis	707/507
5,805,802	9/1998	Marx	380/4
5,845,070	12/1998	Ikudome	380/25
5,895,454	4/1999	Harrington	705/26
5,897,622	4/1999	Blinn et al.	705/26
5,898,777	4/1999	Tycksen, Jr. et al.	380/4
5,909,492	6/1999	Payne et al.	380/24
5,918,213	6/1999	Bernard et al.	705/26
5,940,807	8/1999	Purcell	705/26

OTHER PUBLICATIONS

T. Berners–Lee and D. Connolly, "Hypertext Markup Language–2.0," RFC 1866, MIT/W3C, Nov., 1995.

J. O'Donnell et al., "Special Edition Using Microsoft Internet Explorer 3," *QUE Corp.*, Table of Contents, 1996.

Schneier, Bruce, "Applied Cryptography," John Wiley & Sons, Inc., Table of Contents, 1994.

Digital's EDI Services, Jul. 26, 1997.

Patterson, Wayne, "Mathematical Cryptology for Computer Scientists and Mathematicians," Rowman & Littlefield, 1987. Table of Contents.

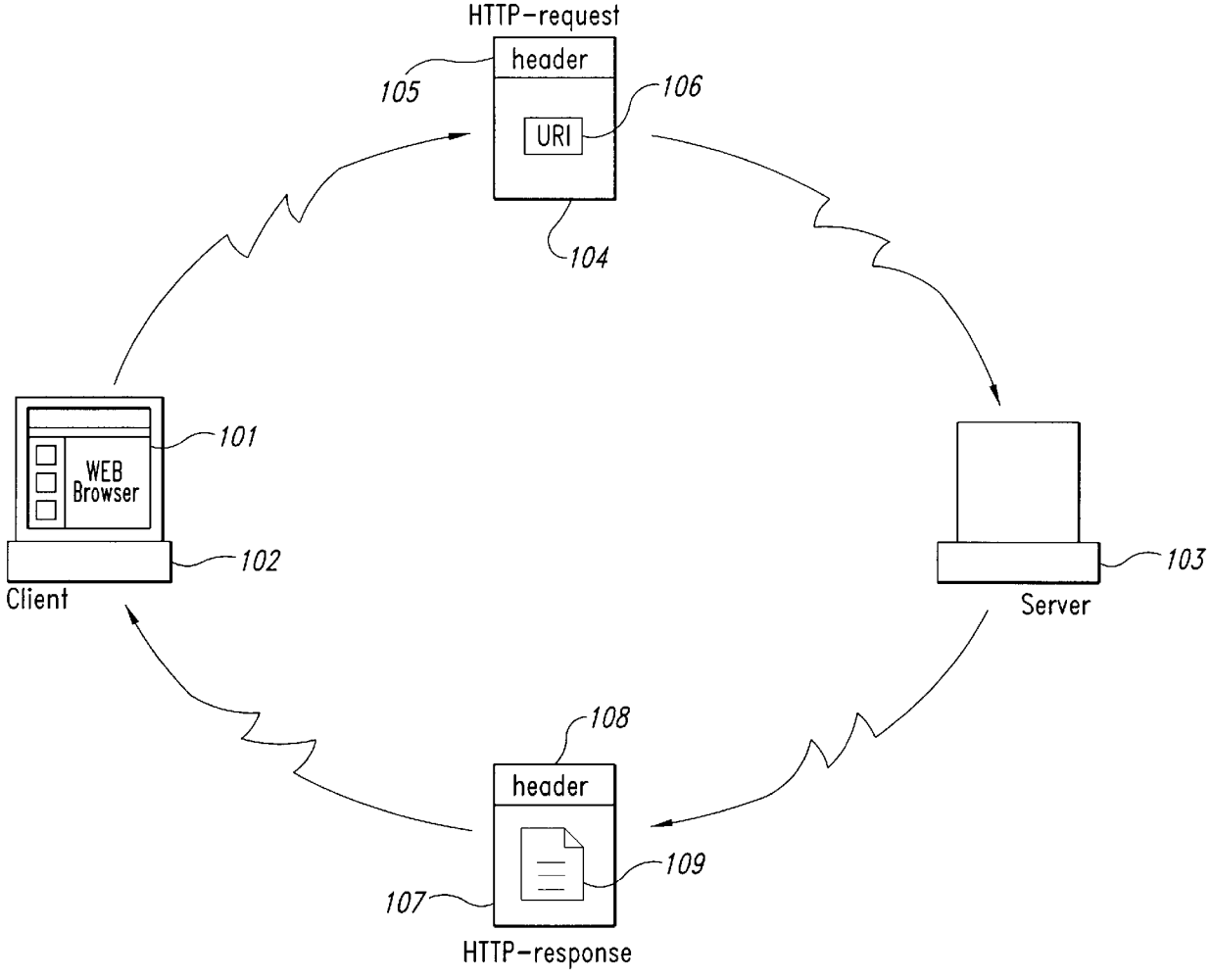

HTTP-request
header
URI
105
106
104
WEB Browser
101
102
Client
103
Server
108
header
109
107
HTTP-response

Fig. 1

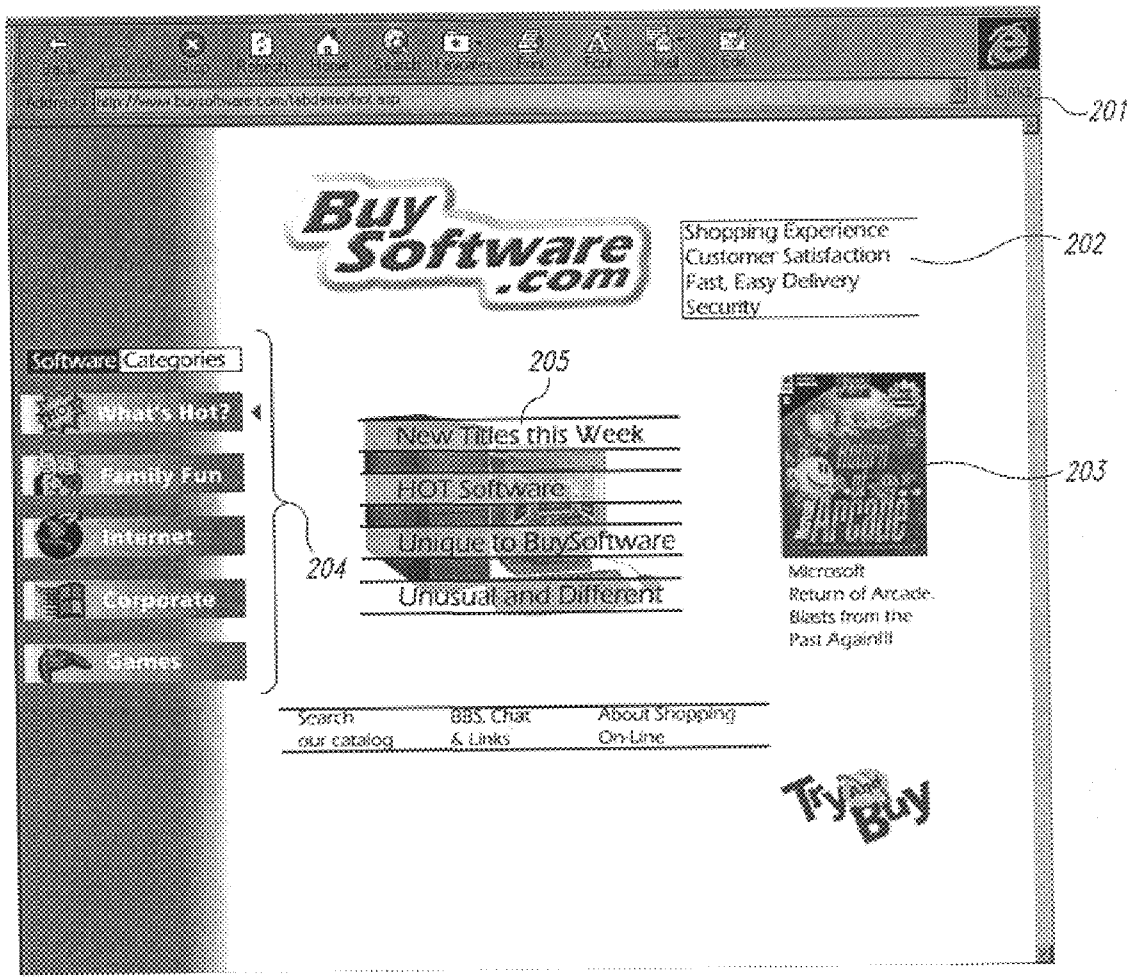

201
BuySoftware.com
Shopping Experience
Customer Satisfaction
Fast, Easy Delivery
Security
202
205
Software Categories
What's Hot?
Family Fun
Internet
Corporate
Games
204
New Titles this Week
Hot Software
Unique to BuySoftware
Unusual and Different
203
Microsoft Return of Arcade Blasts from the Past Again!!!
Search our catalog
BBS, Chat & Links
About Shopping On-Line
Try Buy

Fig. 2

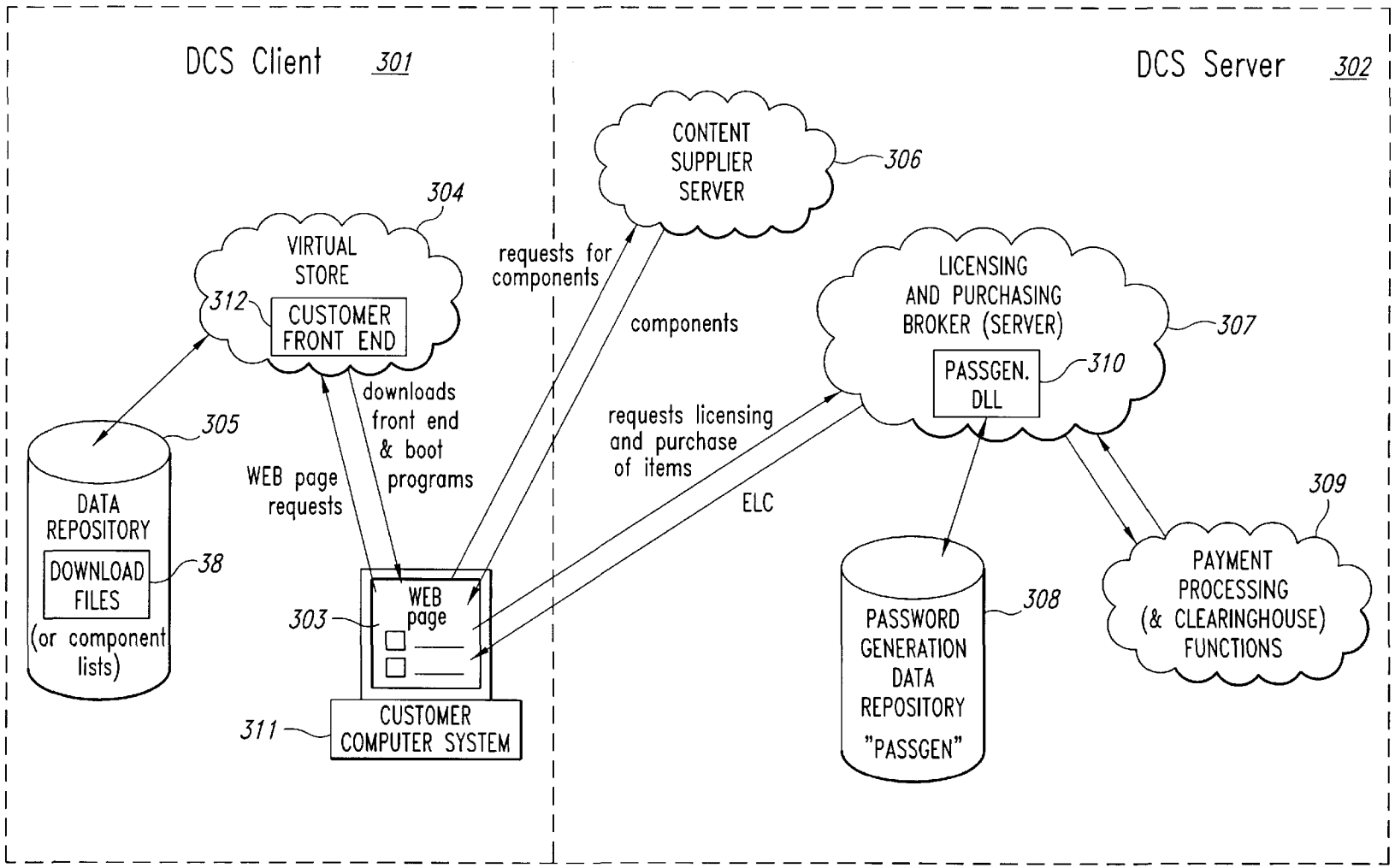

DCS Client 301
DCS Server 302
CONTENT SUPPLIER SERVER
306
304
VIRTUAL STORE
312
CUSTOMER FRONT END
requests for components
components
LICENSING AND PURCHASING BROKER (SERVER)
307
PASSGEN. DLL
310
downloads front end & boot programs
requests licensing and purchase of items
ELC
305
DATA REPOSITORY
DOWNLOAD FILES
38
(or component lists)
WEB page requests
303
WEB page
311
CUSTOMER COMPUTER SYSTEM
PASSWORD GENERATION DATA REPOSITORY "PASSGEN"
308
309
PAYMENT PROCESSING (& CLEARINGHOUSE) FUNCTIONS

Fig. 3

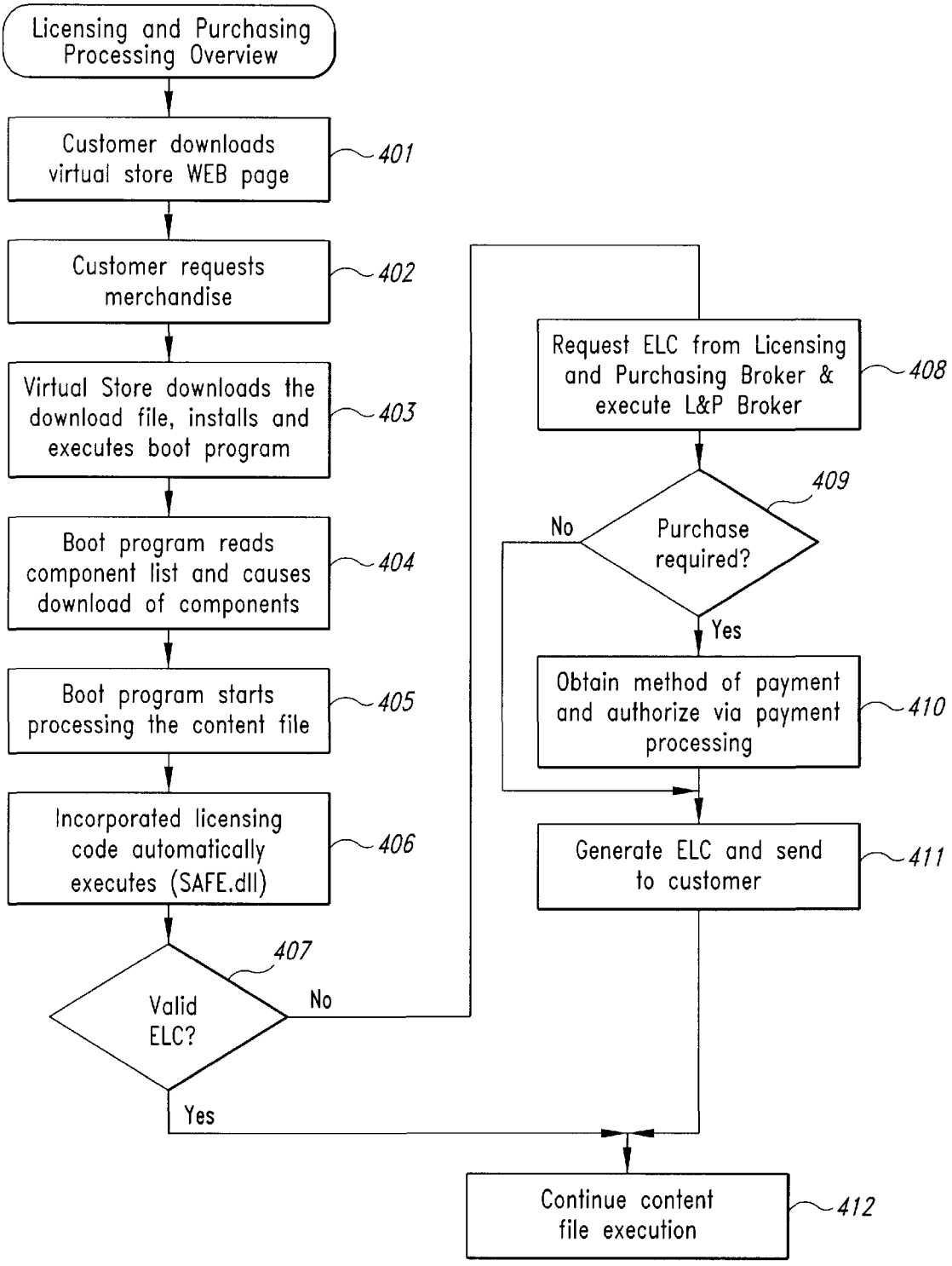

Licensing and Purchasing Processing Overview
Customer downloads virtual store WEB page
401
Customer requests merchandise
402
Virtual Store downloads the download file, installs and executes boot program
403
Boot program reads component list and causes download of components
404
Boot program starts processing the content file
405
Incorporated licensing code automatically executes (SAFE.dll)
406
Valid ELC?
407
No
Yes
Request ELC from Licensing and Purchasing Broker & execute L&P Broker
408
Purchase required?
409
No
Yes
Obtain method of payment and authorize via payment processing
410
Generate ELC and send to customer
411
Continue content file execution
412

Fig. 4

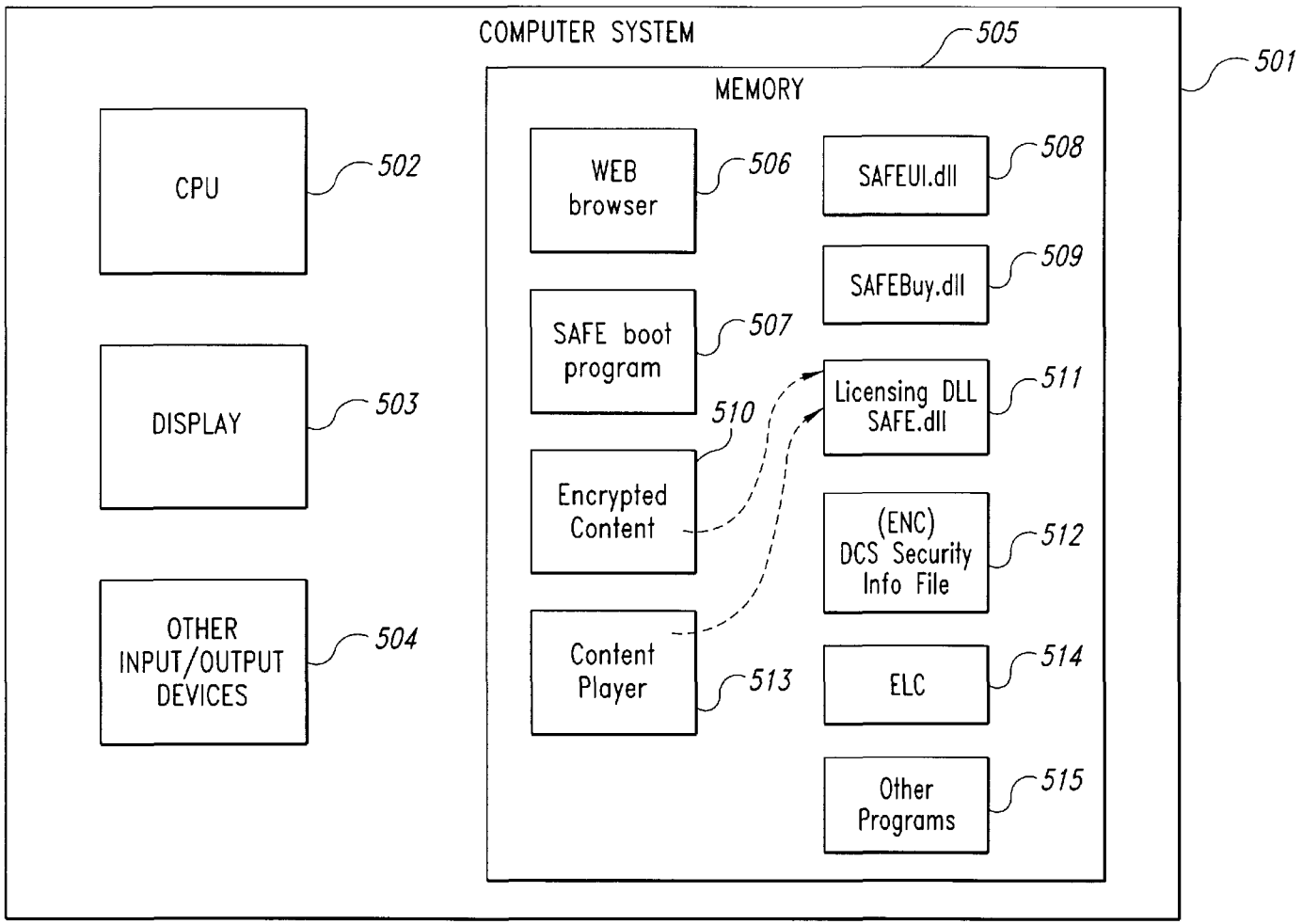

COMPUTER SYSTEM
501
505
MEMORY
CPU
502
DISPLAY
503
OTHER INPUT/OUTPUT DEVICES
504
WEB browser
506
SAFE boot program
507
Encrypted Content
510
Content Player
513
SAFEUI.dll
508
SAFEBuy.dll
509
Licensing DLL SAFE.dll
511
(ENC) DCS Security Info File
512
ELC
514
Other Programs
515

Fig. 5

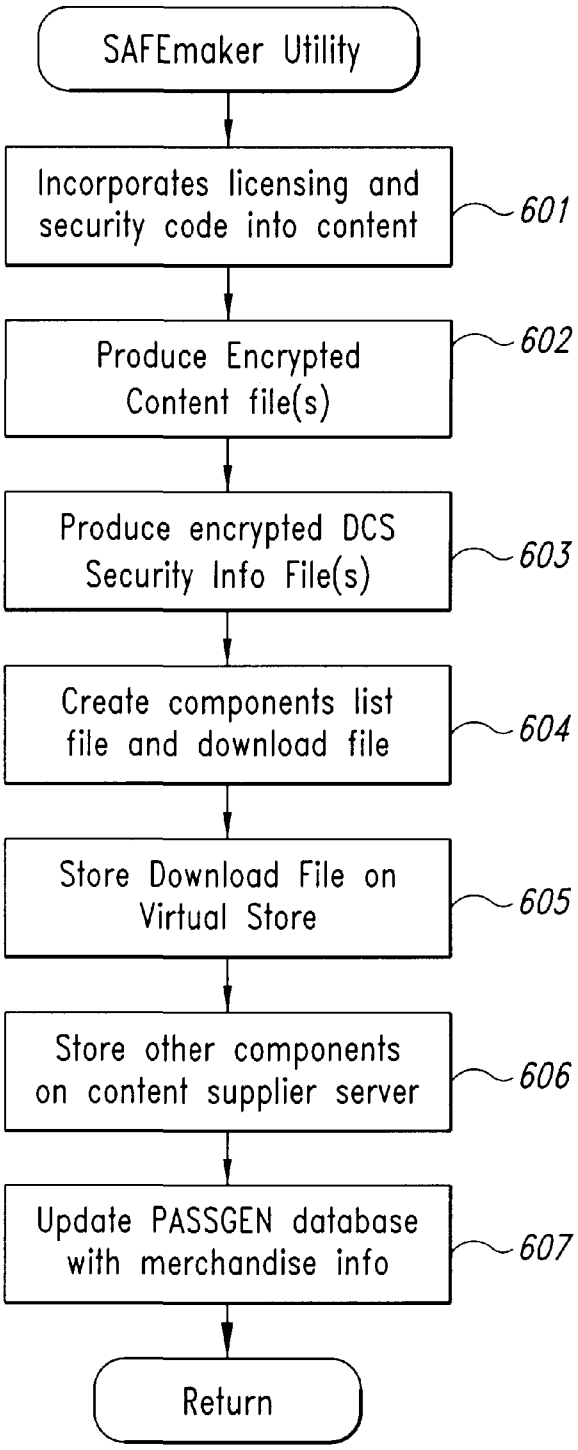

SAFEmaker Utility
Incorporates licensing and security code into content
601
Produce Encrypted Content file(s)
602
Produce encrypted DCS Security Info File(s)
603
Create components list file and download file
604
Store Download File on Virtual Store
605
Store other components on content supplier server
606
Update PASSGEN database with merchandise info
607
Return

Fig. 6

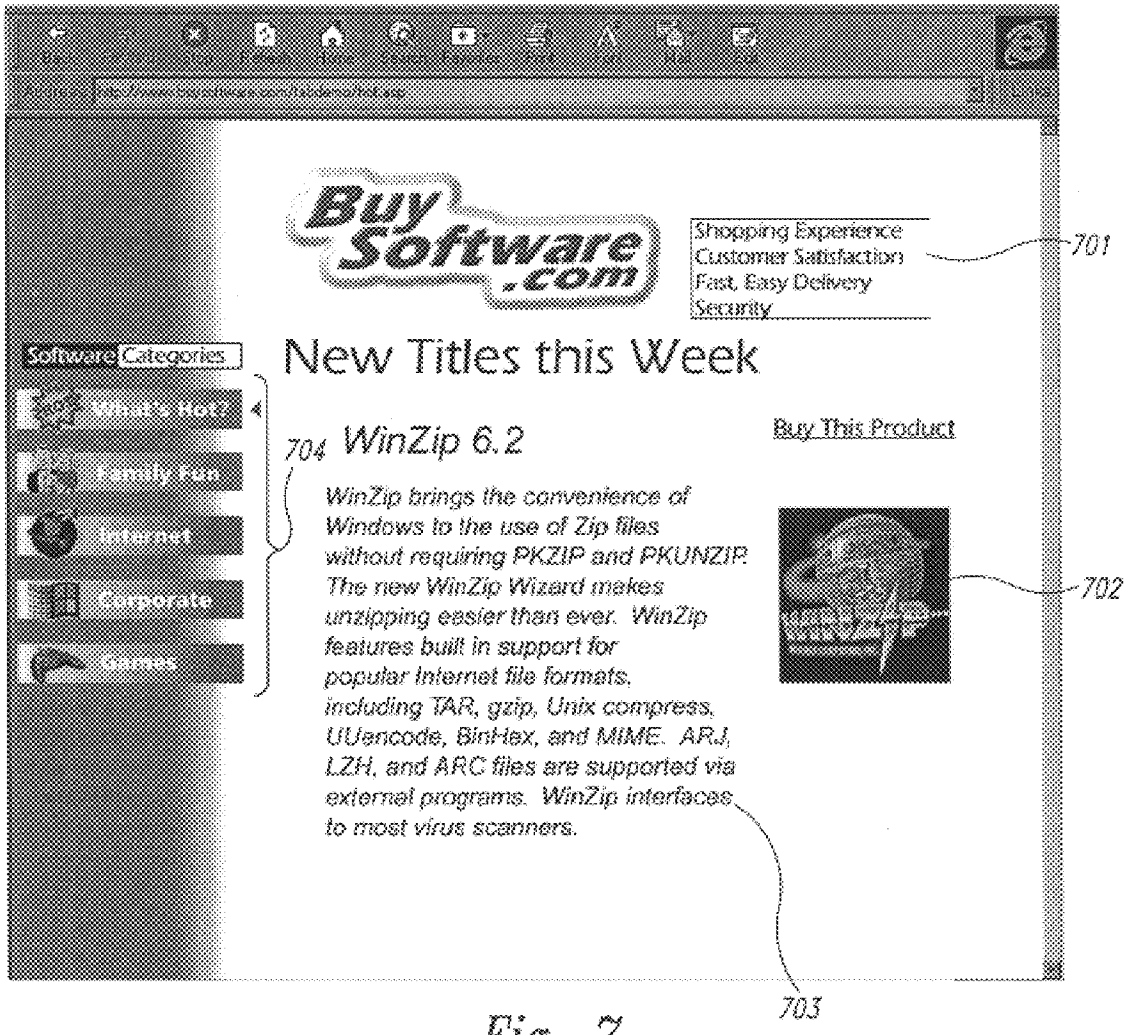

Buy Software .com
Shopping Experience
Customer Satisfaction
Fast, Easy Delivery
Security
701
Software Categories
What's Hot?
Family Fun
Internet
Corporate
Games
New Titles this Week
704
WinZip 6.2
Buy This Product
WinZip brings the convenience of Windows to the use of Zip files without requiring PKZIP and PKUNZIP. The new WinZip Wizard makes unzipping easier than ever. WinZip features built in support for popular Internet file formats, including TAR, gzip, Unix compress, UUencode, BinHex, and MIME. ARJ, LZH, and ARC files are supported via external programs. WinZip interfaces to most virus scanners.
702
703

Fig. 7

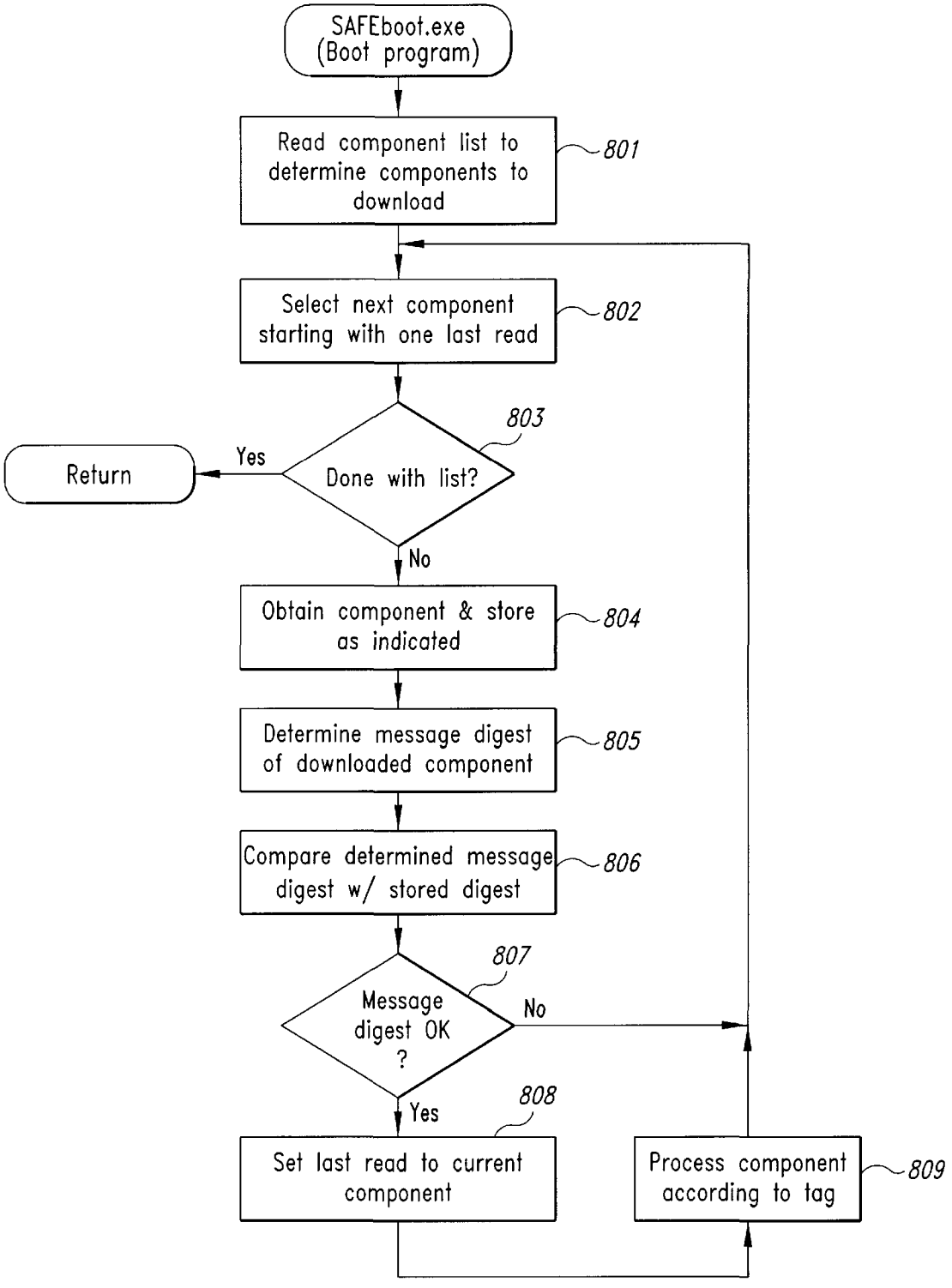

SAFEboot.exe
(Boot program)
Read component list to determine components to download
801
Select next component starting with one last read
802
803
Done with list?
Yes
Return
No
Obtain component & store as indicated
804
Determine message digest of downloaded component
805
Compare determined message digest w/ stored digest
806
807
Message digest OK ?
No
Yes
808
Set last read to current component
Process component according to tag
809

Fig. 8

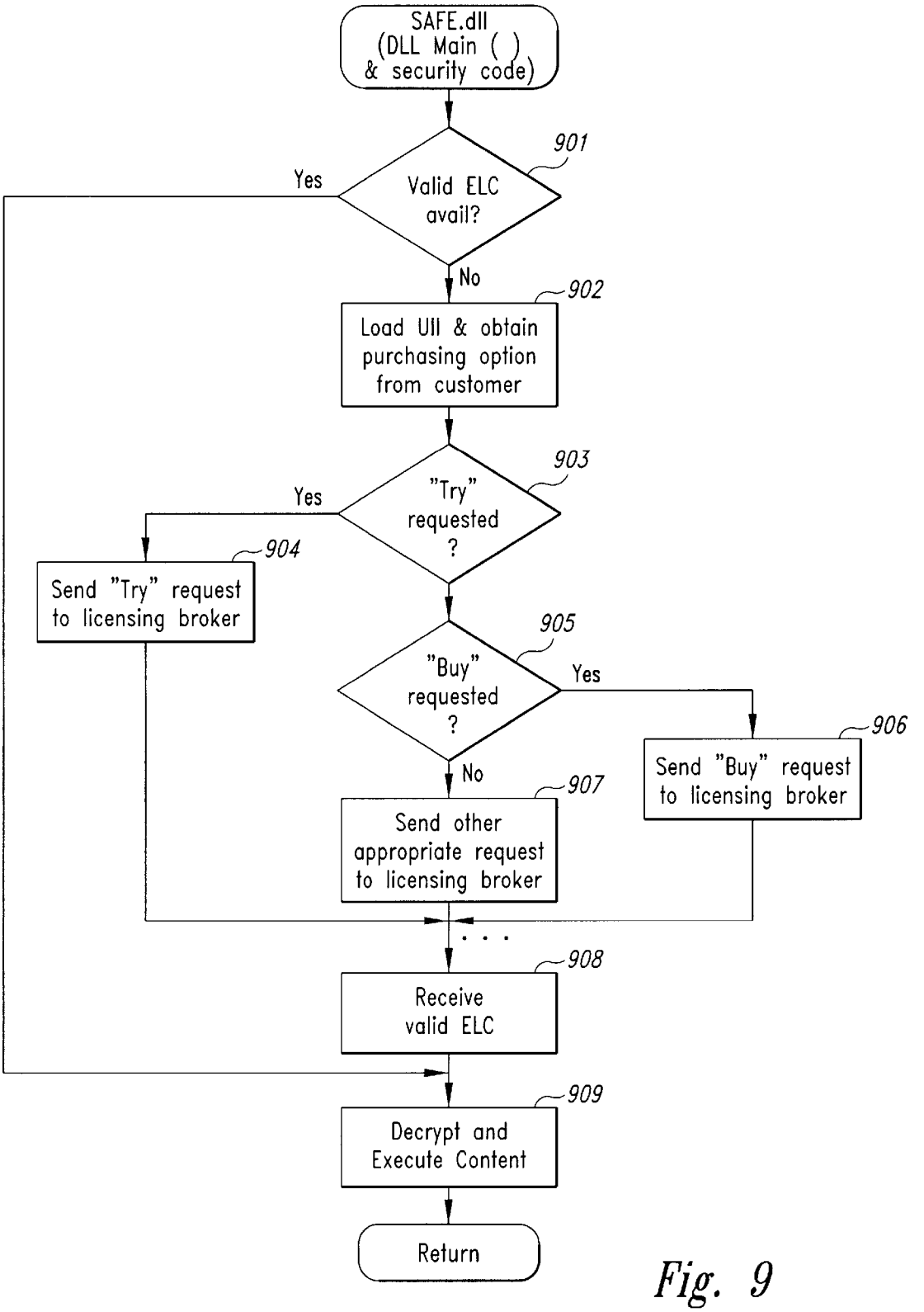

SAFE.dll
(DLL Main ( )
& security code)
901
Valid ELC
avail?
Yes
No
902
Load UII & obtain
purchasing option
from customer
903
"Try"
requested
?
Yes
904
Send "Try" request
to licensing broker
905
"Buy"
requested
?
Yes
906
Send "Buy" request
to licensing broker
No
907
Send other
appropriate request
to licensing broker
908
Receive
valid ELC
909
Decrypt and
Execute Content
Return

Fig. 9

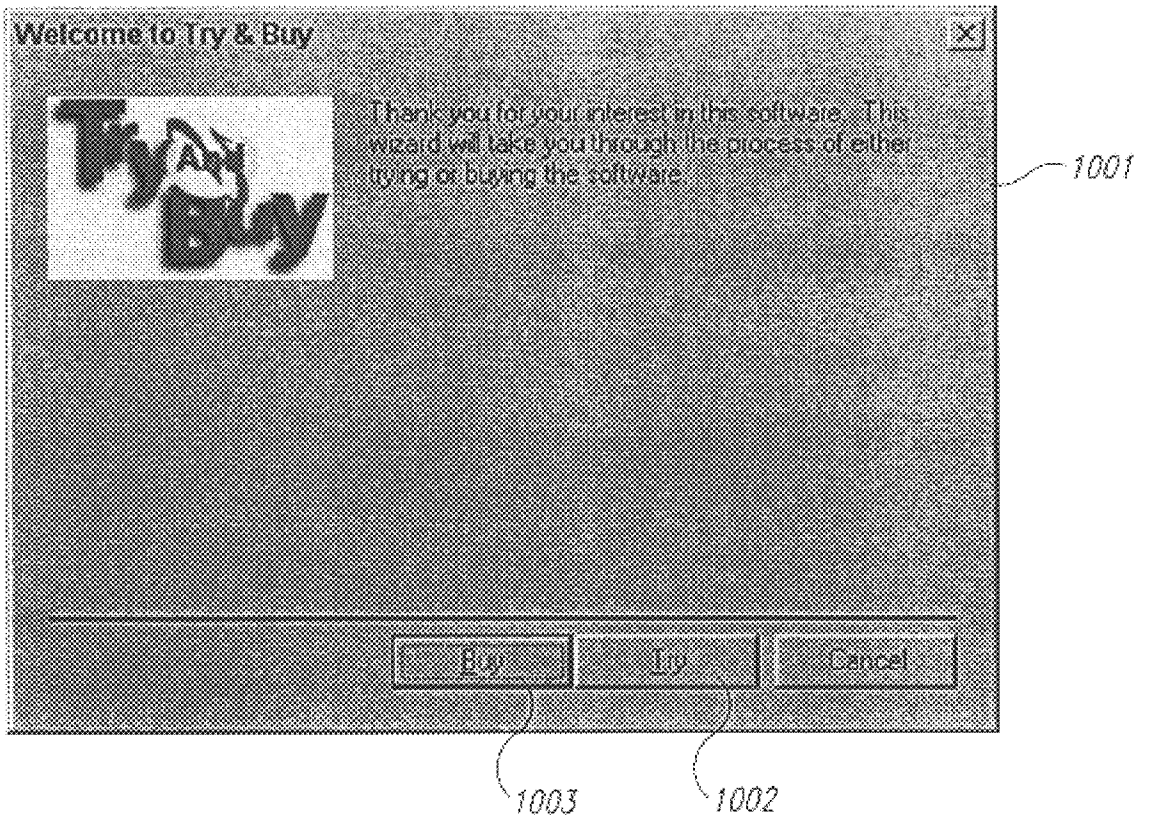

Welcome to Try & Buy
Thank you for your interest in the software. This wizard will take you through the process of either trying or buying the software.
Try
Buy
Cancel
1001
1003
1002

Fig. 10

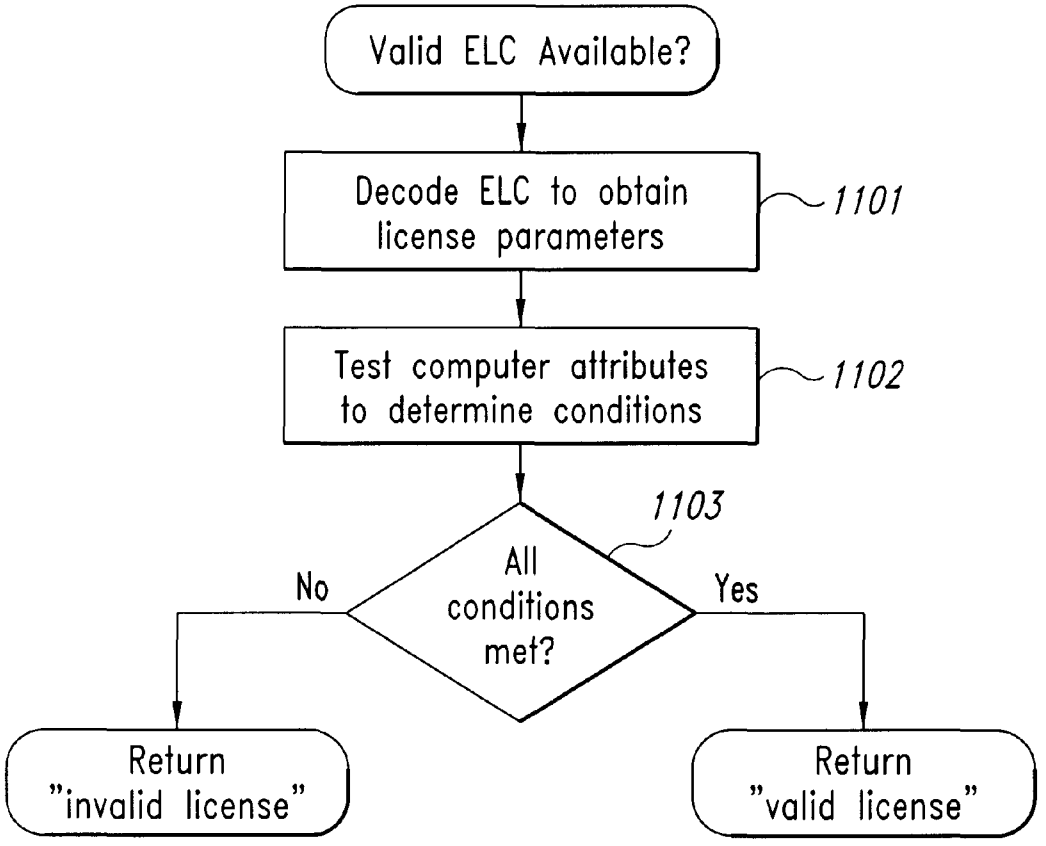

Valid ELC Available?
Decode ELC to obtain license parameters
1101
Test computer attributes to determine conditions
1102
1103
All conditions met?
No
Yes
Return "invalid license"
Return "valid license"

Fig. 11

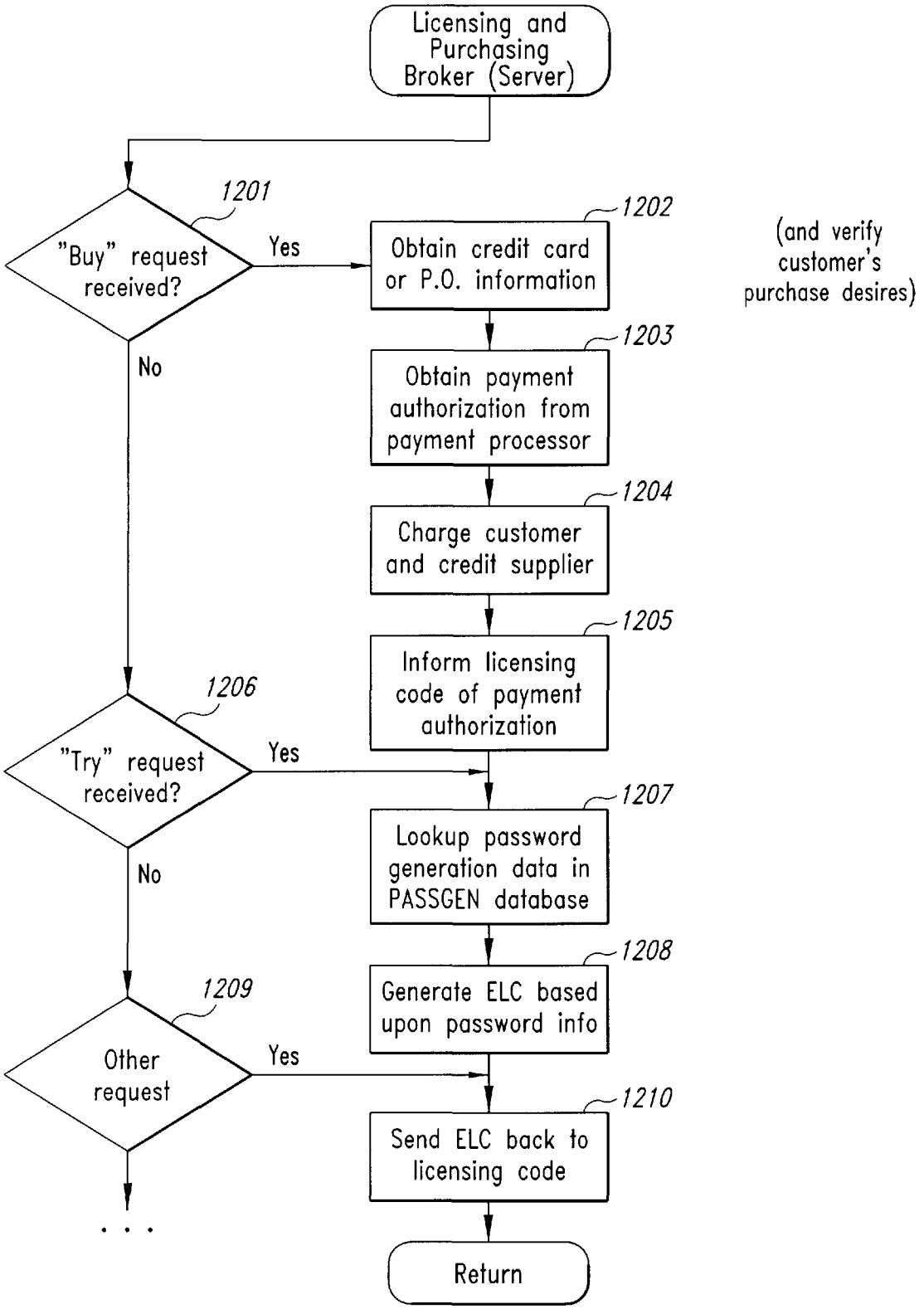

Licensing and Purchasing Broker (Server)
1201
"Buy" request received?
Yes
No
1202
Obtain credit card or P.O. information
(and verify customer's purchase desires)
1203
Obtain payment authorization from payment processor
1204
Charge customer and credit supplier
1205
Inform licensing code of payment authorization
1206
"Try" request received?
Yes
No
1207
Lookup password generation data in PASSGEN database
1208
Generate ELC based upon password info
1209
Other request
Yes
1210
Send ELC back to licensing code
. . .
Return

Fig. 12

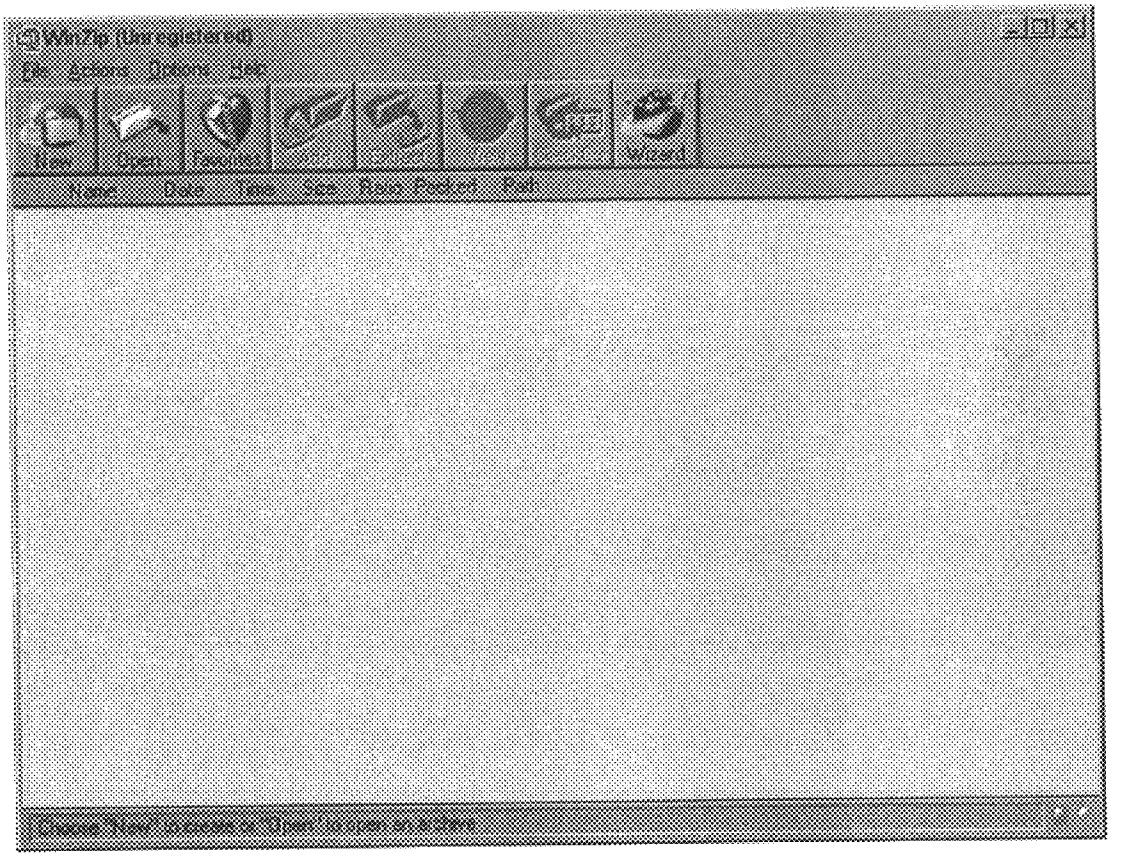


Fig. 13

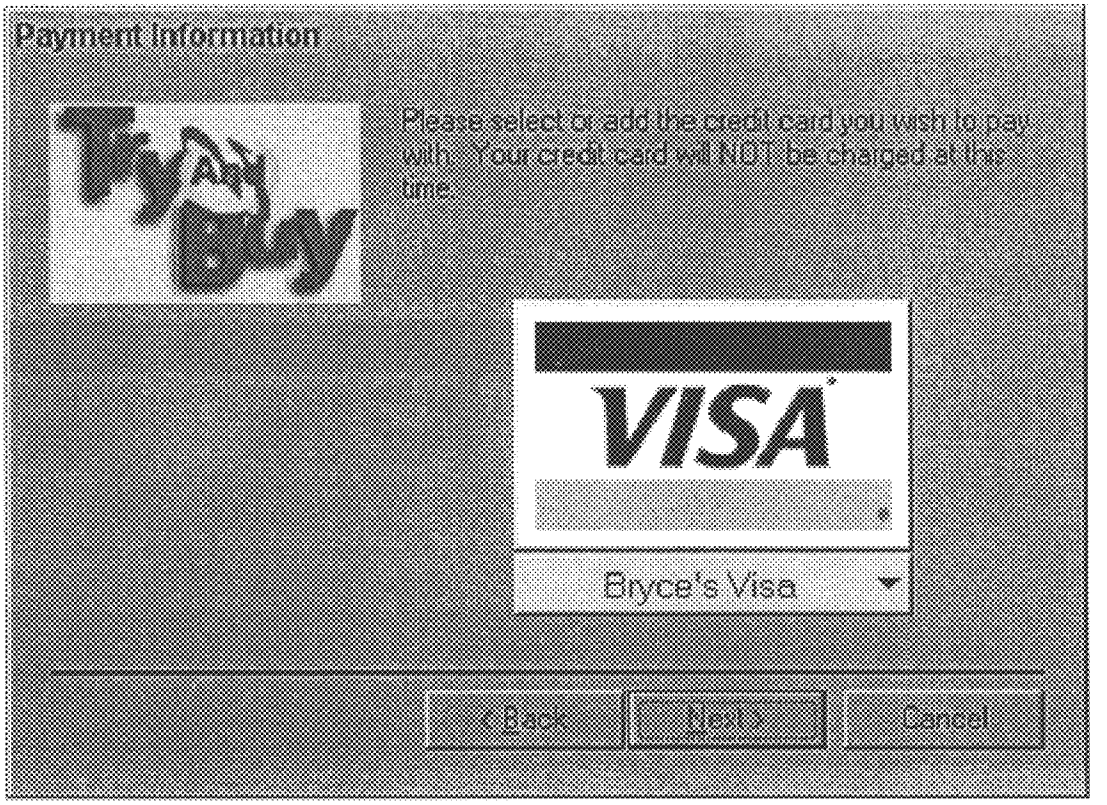

Payment Information
Please select or add the credit card you wish to pay with. Your credit card will NOT be charged at this time.
VISA
Bryce's Visa
< Back
Next >
Cancel

Fig. 14

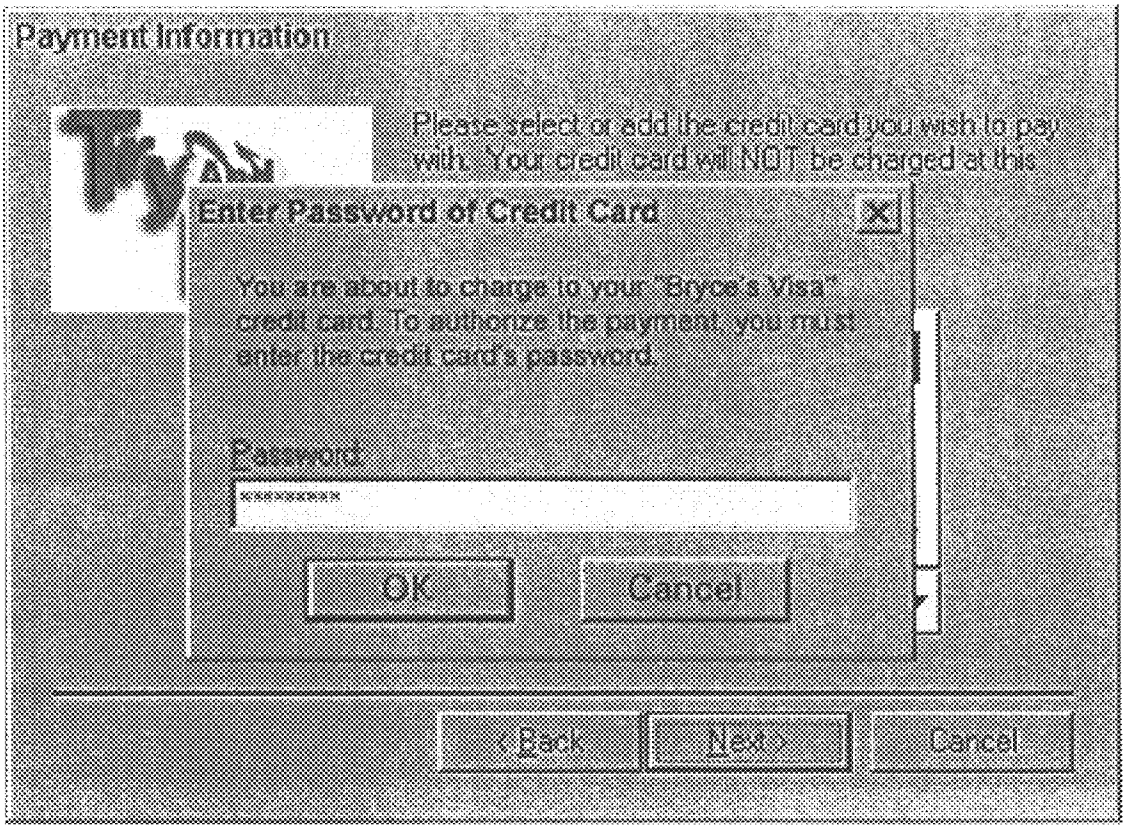

Payment Information
Please select or add the credit card you wish to pay with. Your credit card will NOT be charged at this
Enter Password of Credit Card
You are about to charge to your "Bryce's Visa" credit card. To authorize the payment, you must enter the credit card's password.
Password:
OK
Cancel
< Back
Next >
Cancel

Fig. 15

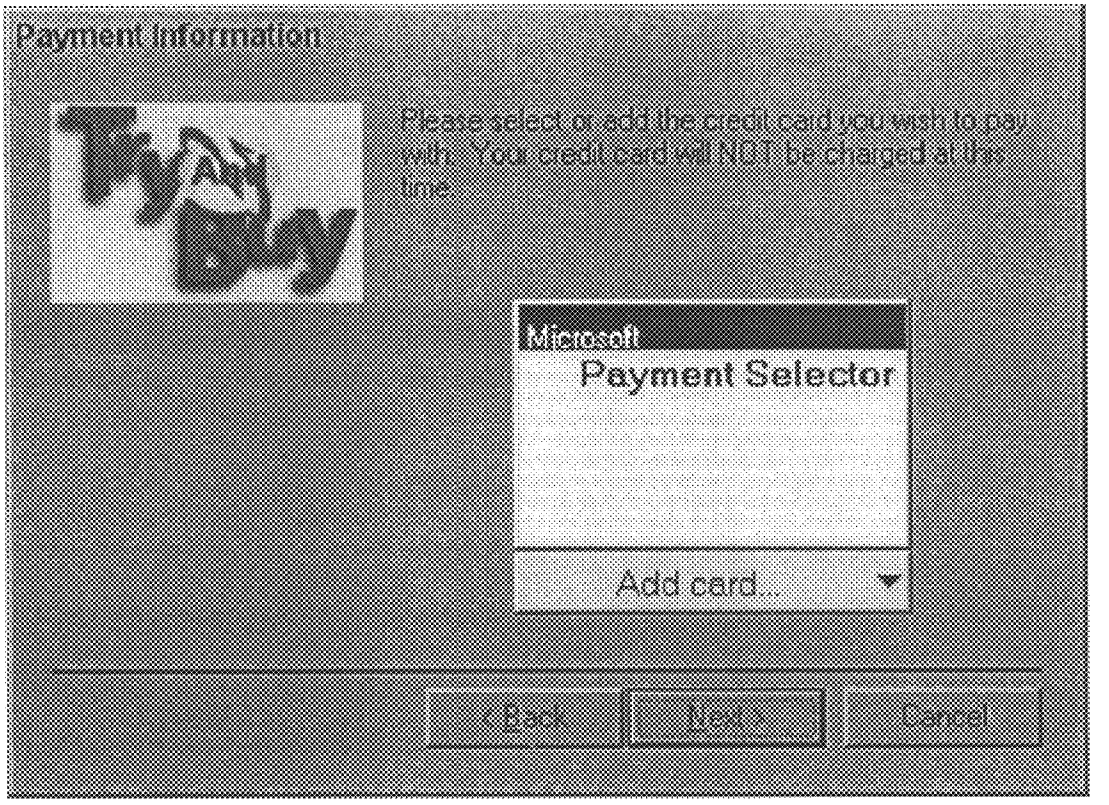

Payment Information
Please select or add the credit card you wish to pay with. Your credit card will NOT be charged at this time.
Microsoft
Payment Selector
Add card...
< Back
Next >
Cancel

Fig. 16

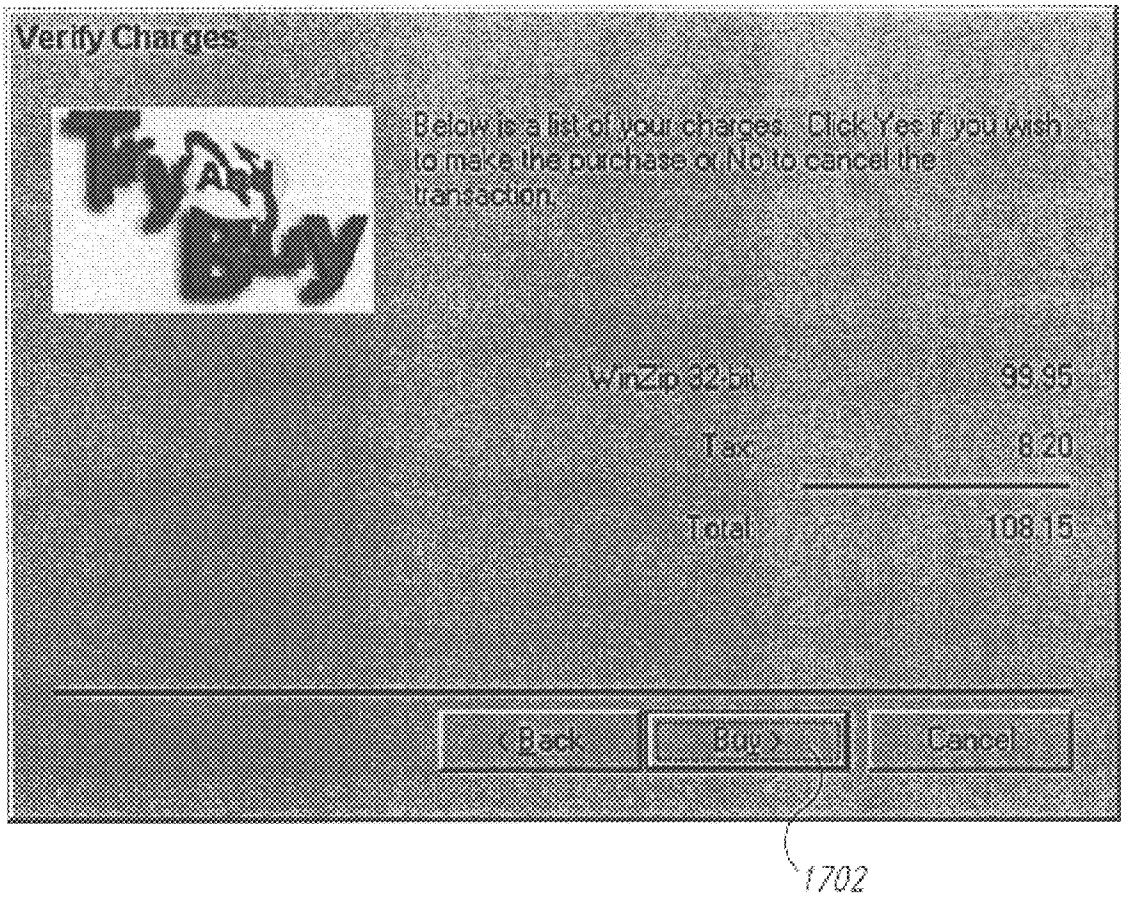

Verify Charges
Below is a list of your charges. Click Yes if you wish to make the purchase or No to cancel the transaction.
WinZip 32-bit
99.95
Tax
8.20
Total
108.15
< Back
Buy >
Cancel
1702

Fig. 17

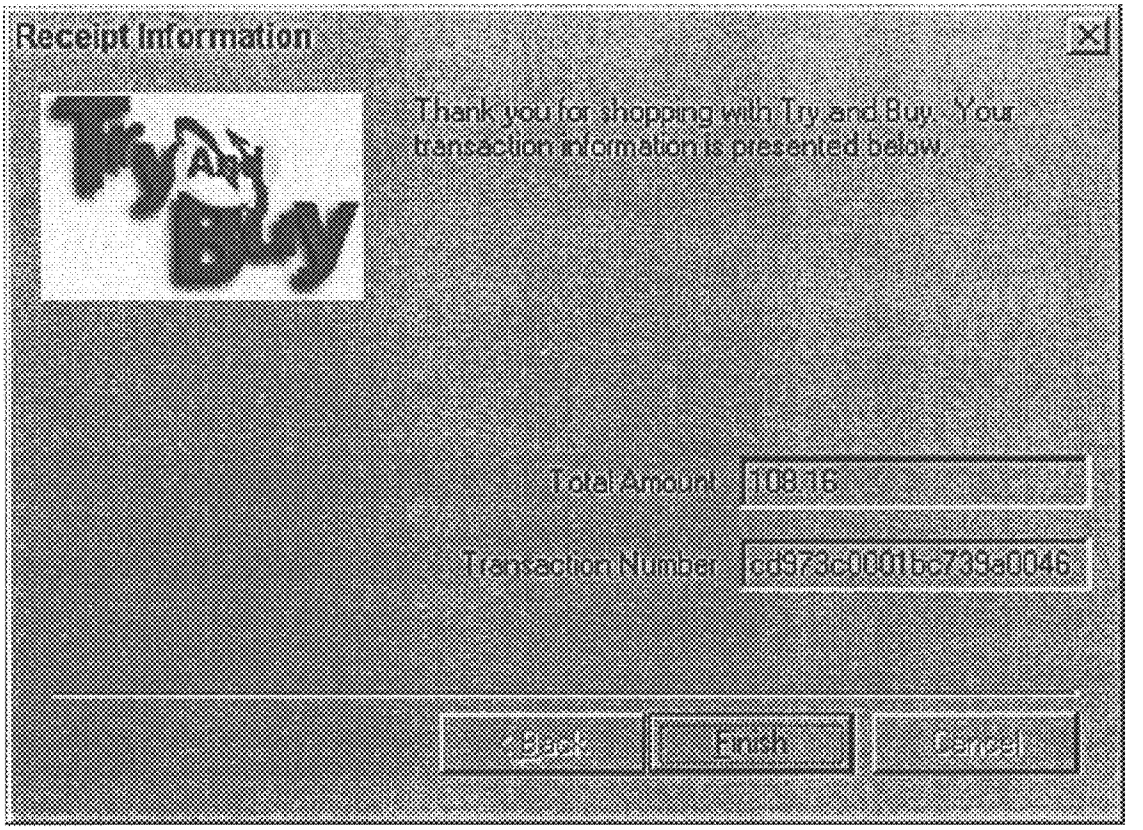

Receipt Information
Thank you for shopping with Try and Buy. Your transaction information is presented below.
Total Amount:
108.16
Transaction Number:
< Back
Finish
Cancel

Fig. 18

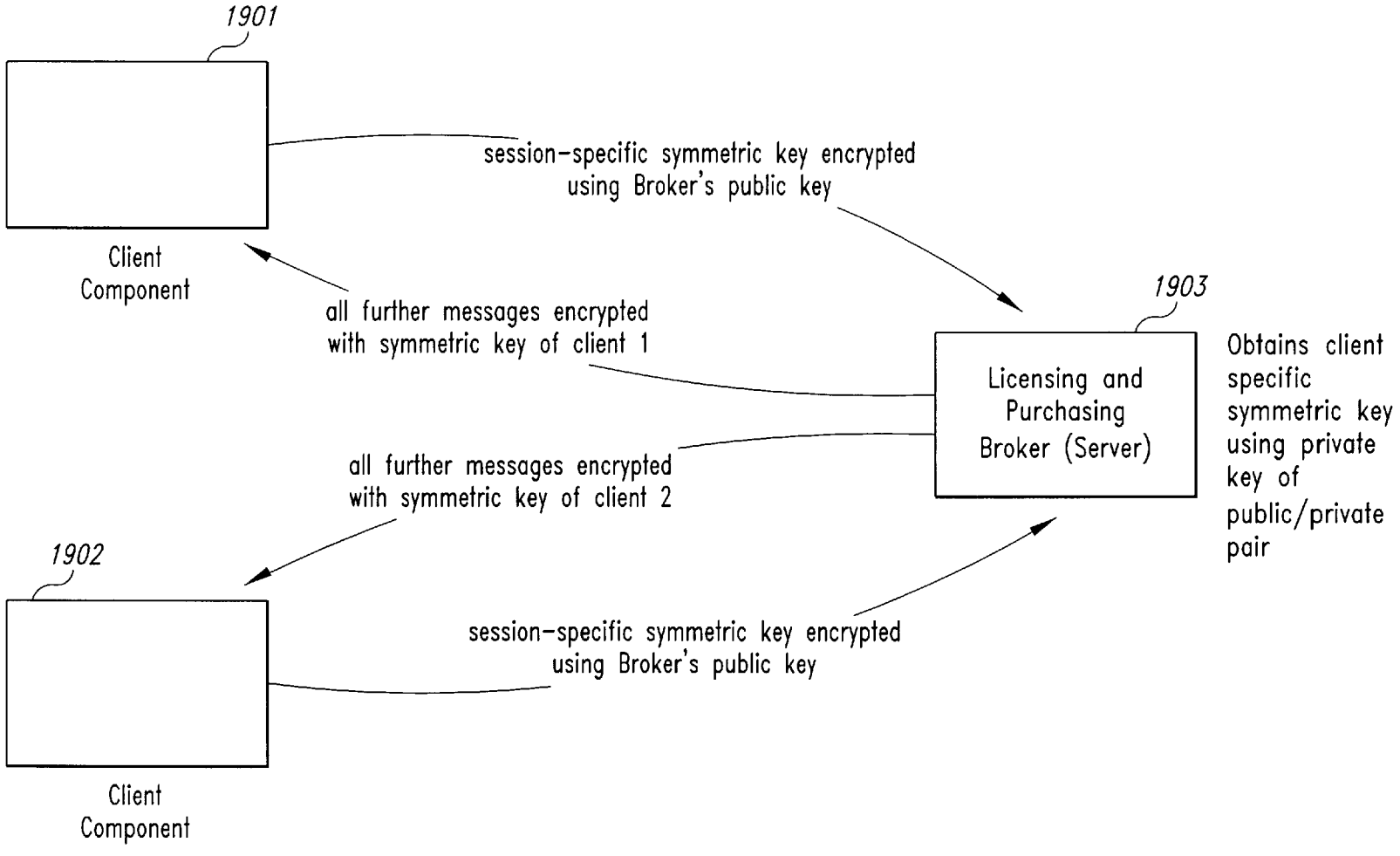
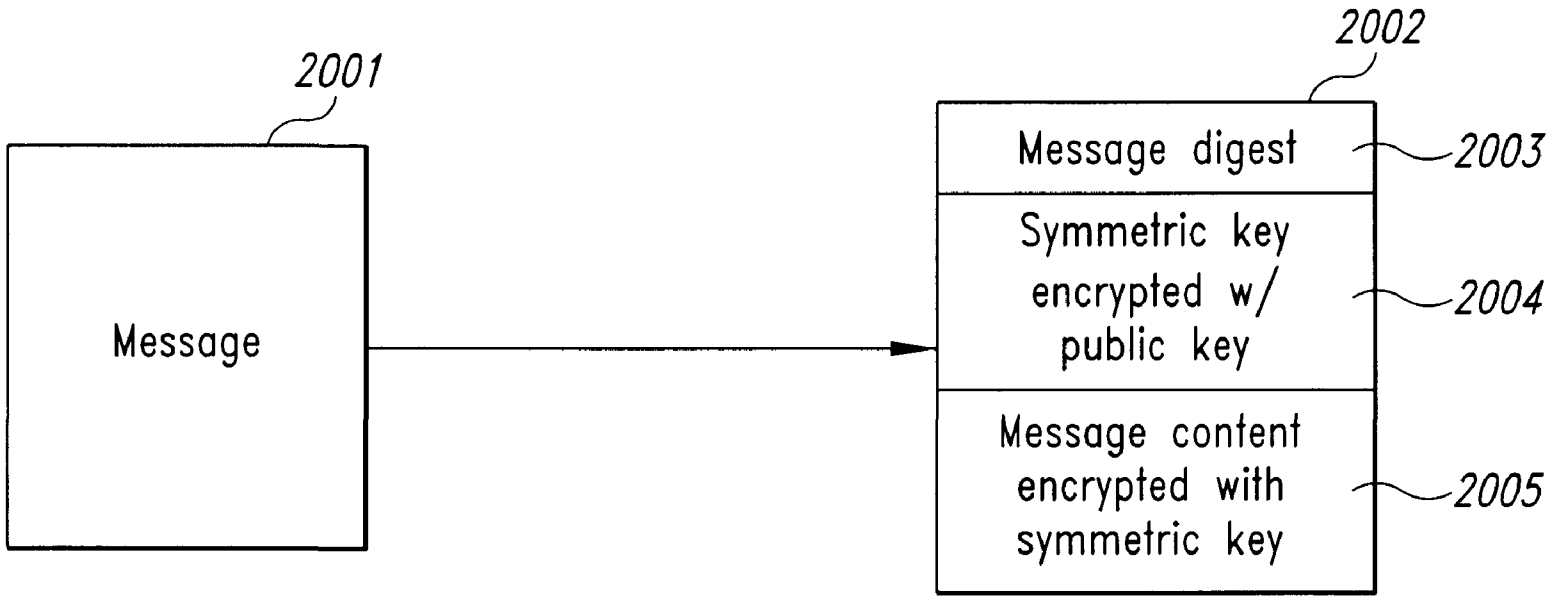

1901
session-specific symmetric key encrypted using Broker's public key
Client Component
all further messages encrypted with symmetric key of client 1
1903
Licensing and Purchasing Broker (Server)
Obtains client specific symmetric key using private key of public/private pair
all further messages encrypted with symmetric key of client 2
1902
session-specific symmetric key encrypted using Broker's public key
Client Component

Fig. 19



Encrypted Message Protocol

Fig. 20

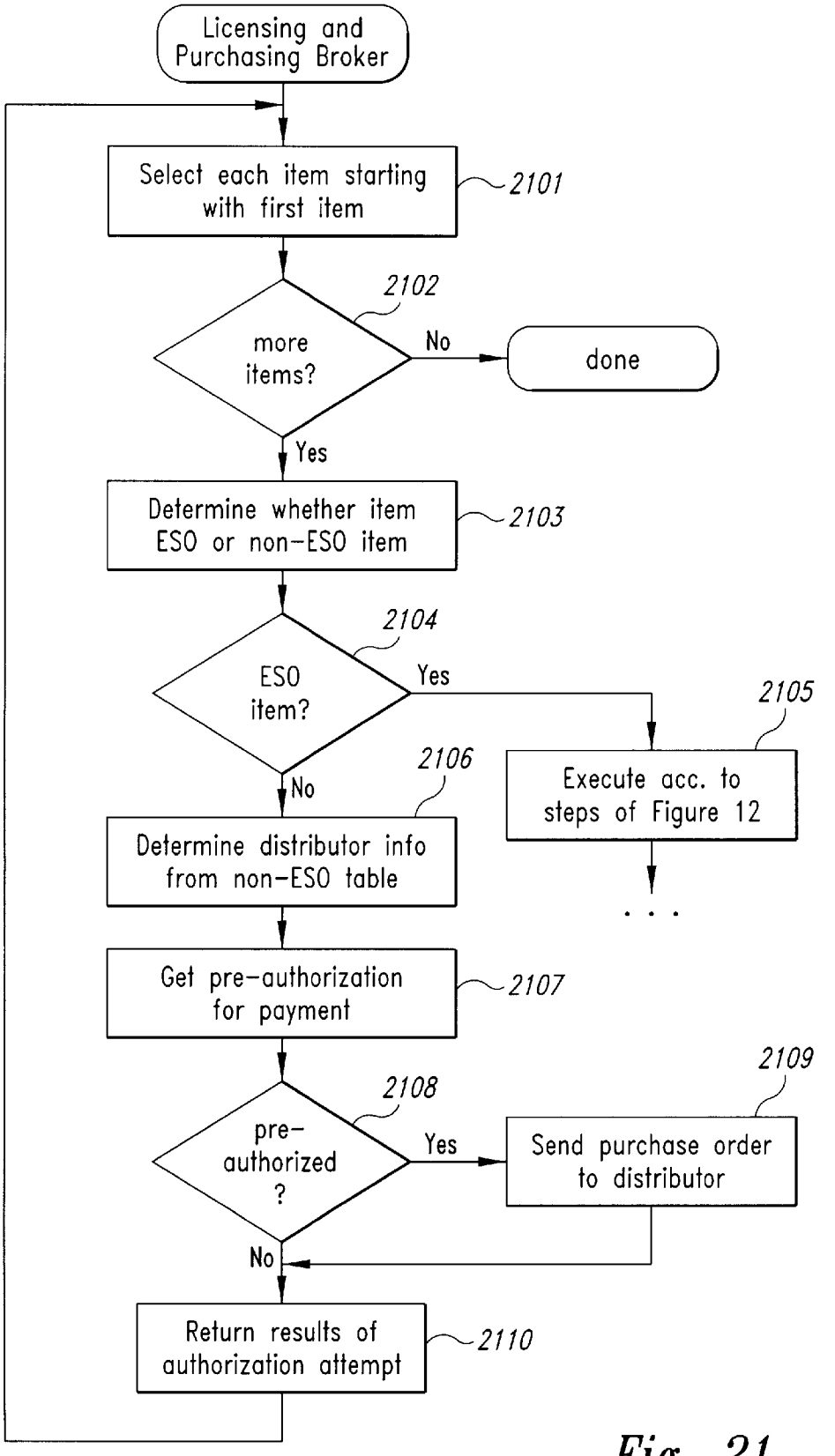

Licensing and Purchasing Broker
Select each item starting with first item
2101
2102
more items?
No
done
Yes
Determine whether item ESO or non-ESO item
2103
2104
ESO item?
Yes
2105
Execute acc. to steps of Figure 12
2106
No
Determine distributor info from non-ESO table
Get pre-authorization for payment
2107
2108
pre-authorized ?
Yes
2109
Send purchase order to distributor
No
Return results of authorization attempt
2110

Fig. 21

METHOD AND SYSTEM FOR SECURELY INCORPORATING ELECTRONIC INFORMATION INTO AN ONLINE PURCHASING APPLICATION

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation-in-part of a U.S. Provisional Application No. 60/049,844, entitled "A Method and System of Securely Incorporating Digital Information into an Electronic Store," filed on Jun. 17, 1997, which is hereby incorporated by reference in its entirety. This application is also a continuation-in-part of U.S. patent application Ser. No. 08/792,719, entitled "Method and System for Injecting New Code Into Existing Application Code," filed on Jan. 29, 1997, and which is hereby incorporated by reference in its entirety.

TECHNICAL FIELD

The present invention relates to facilitating the purchase of electronic information using digital commerce and, in particular, to providing a component-based architecture that facilitates online licensing and purchase of digital content and software.

BACKGROUND OF THE INVENTION

Today's computer networking environments, such as the Internet, offer an unprecedented medium for facilitating the purchase of software and digital content online. Electronic software distribution (ESD) provides an online alternative (using computers) for a customer to purchase software and other types of digital content from publishers, resellers, and distributors without the physical distribution of a shrink-wrapped product. This online process is referred to as digital commerce. The customer purchases and downloads the software or other digital content directly from the network. In the context of this specification, software is generally a computer program, which is self-executing, whereas digital content that is not software is data that serves as input to another computer program. For example, audio content is digital content (an audio script) that is played and heard by executing an audio player (a computer program) to process the audio script. This act of processing is referred to as "executing" the digital content. For the purposes of this specification, self-executing content and other digital content, as well as any other type of electronic information that can be licensed or purchased, including combinations of content and a player for that content, will be referred to generically as electronic information, electronic data, or electronic content.

One of the major problems that authors of electronic content face using digital commerce is a reliable mechanism for obtaining payment for their electronic content. One reason is that it has become increasingly easy, without the use of secure licensing code, to copy and widely distribute electronic content. To limit the use of illegal copies of electronic content, current systems have incorporated licensing code into existing application programs to be electronically distributed using various solutions. According to one technique, which will be referred to herein as "wrapping," a second application program (a wrapper program) is distributed on the network, which includes an encrypted version of the original application program. The wrapper program, when installed, decrypts the encrypted original application program and then proceeds to execute the original application program. To successfully decrypt the program, a legitimate end user must provide the proper licensing information to enable the decryption to operate. A security hole exists, however, in that, while the wrapping program is in the process of decrypting the original application executable file, temporary files are created to hold the decrypted program code. Once the entire original application program has been decrypted and stored in the temporary file, a "software pirate" can then make multiple copies of the original unencrypted application program in the temporary file and can distribute them illegally.

Further, use of the wrapping technique to incorporate licensing provides only limited additional security to a vendor who implements what is known as a "try and buy" licensing model. A try and buy licensing model typically distributes an application program with either limited functionality or for a limited time of use to enable a potential customer to explore the application. Functionality may be limited, for example, by disabling a set of features. Once the potential customer is satisfied, the customer can pay for and license the application program for more permanent use. If an application program is distributed using the wrapping technique to potential customers for the purpose of try and buy licensing, then, when the application program is decrypted and stored in a temporary file, a software pirate can determine how to enable the disabled features or how to remove the license expiration data. These security problems can result in the distribution of illegal copies, which are hard to detect and monitor in a global network environment.

A second technique for incorporating licensing code into an existing application program directly inserts the licensing code into the executable file. Using the direct insertion method, an application developer determines where in the executable file the licensing code should be placed and inserts the new code into the executable. After inserting the licensing code into the existing executable file, the application developer adjusts addresses that reference any relocatable code or data that follows the inserted code to account for the newly added code. However, it is very difficult for an application developer to determine where to insert the licensing code and to then test the entire application to ensure it works correctly. An application developer would typically need to disassemble the executable file and study the disassembled code to determine where to insert the licensing code. Such disassembling and studying is a very time-consuming process. Furthermore, the process must be repeated for each application program, and for each version of each application program in which the code is to be inserted.

In addition to problems relating to obtaining payment due to illegal distribution, the current methods for incorporating licensing code and for supporting digital commerce present scalability problems. For example, it is difficult for these systems to handle large volumes and numerous types of electronic content because any change to the licensing or purchasing model requires re-encryption and perhaps re-wrapping of the electronic content. In addition, it is difficult to distribute such content online when the content is large in size because the network connection may be prone to failures. A failure in a network connection when downloading the electronic content would require starting the download operation again.

To perform digital commerce, today's computer networking environments utilize a client/server architecture and a standard protocol for communicating between various network sites. One such network, the World Wide WEB network, which comprises a subset of Internet sites, supports a standard protocol for requesting and for receiving docu-

ments known as WEB pages. This protocol is known as the Hypertext Transfer Protocol, or "HTTP." HTTP defines a high-level message passing protocol for sending and receiving packets of information between diverse applications. Details of HTTP can be found in various documents including T. Berners-Lee et al., *Hypertext Transfer Protocol—HTTP* 1.0, Request for Comments (RFC) 1945, MIT/LCS, May, 1996, which is incorporated herein by reference. Each HTTP message follows a specific layout, which includes among other information a header, which contains information specific to the request or response. Further, each HTTP message that is a request (an HTTP request message) contains a universal resource identifier (a "URI"), which specifics a target network resource for the request. A URI is either a Uniform Resource Locator ("URL") or Uniform Resource Name ("URN"), or any other formatted string that identifies a network resource. The URI contained in a request message, in effect, identifies the destination machine for a message. URIs, as an example of URIs, are discussed in detail in T. Berners-Lee, et al., *Uniform Resource Locators* (URL), RFC 1738, CERN, Xerox PARC, Univ. of Minn., December, 1994, which is incorporated herein by reference.

FIG. **1** illustrates how a browser application, using the client/server model of the World Wide WEB network, enables users to navigate among network nodes by requesting and receiving WEB pages. For the purposes of this specification, a WEB page is any type of document that abides by the HTML format. That is, the document includes an "<HTML>" statement. Thus, a WEB page can also be referred to as an HTML document or an HTML page. HTML is a document mark-up language, defined by the Hypertext Markup Language ("HTML") specification. HTML defines tags for specifying how to interpret the text and images stored in an HTML page. For example, there are HTML tags for defining paragraph formats and text attributes such as boldface and underlining. In addition, the HTML format defines tags for adding images to documents and for formatting and aligning text with respect to images. HTML tags appear between angle brackets, for example, <HTML>. Further details of HTML arc discussed in T. Berners-Lee and D. Connolly, *Hypertext Markup Language*-2.0, RFC 1866, MIT/W3C, November, 1995, which is incorporated herein by reference.

In FIG. **1**, a WEB browser application **101** is shown executing on a client computer system **102**, which communicates with a server computer system **103** by sending and receiving HTTP packets (messages). The WEB browser application **101** requests WEB pages from other locations on the network to browse (display) what is available at these locations. This process is known as "navigating" to sites on the WEB network. In particular, when the WEB browser application **101** "navigates" to a new location, it requests a new page from the new location (e.g., server computer system **103**) by sending an HTTP-request message **104** using any well-known underlying communications wire protocol. HTTP-request message **104** follows the specific layout discussed above, which includes a header **105** and a URI field **106**, which specifies the target network location for the request. When the server computer system machine specified by URI **106** (e.g., the server computer system **103**) receives the HTTP-request message, it decomposes the message packet and processes the request. When appropriate, the server computer system constructs a return message packet to send to the source location that originated the message (e.g., the client computer system **102**) in the form of an HTTP-response message **107**. In addition to the standard features of an HTTP message, such as the header **108**, the HTTP-response message **107** contains the requested WEB page **109**. When the HTTP-response message **107** reaches the client computer system **102**, the WEB browser application **101** extracts the WEB page **109** from the message, and parses and interprets the HTML code in the page (executes the WEB page) in order to display the document on a display screen of the client computer system **102** in accordance with the HTML tags.

SUMMARY OF THE INVENTION

The present invention provides methods and systems for facilitating the purchase and delivery of electronic content using a secure digital commerce system. The secure digital commerce system interacts with an online purchasing system to purchase and distribute merchandise over a network. The secure digital commerce system is comprised of a plurality of modularized components, which communicate with each other to download, license, and potentially purchase a requested item of merchandise. Each component is customizable.

Exemplary embodiments of the secure digital commerce system ("DCS") include a DCS client and a DCS server. The DCS client includes a plurality of client components, which are downloaded by a boot program onto a customer computer system in response to requesting an item of merchandise to be licensed or purchased. The downloaded client components include a secured (e.g., encrypted) content file that corresponds to the content of the requested item and licensing code that is automatically executed to ensure that the item of merchandise is properly licensed before a customer is permitted to operate it. The DCS server includes a content supplier server, which provides the DCS client components that are specific to the requested item, and a licensing and purchasing broker, which generates and returns a secure electronic licensing certificate in response to a request to license the requested item of merchandise. The generated electronic license certificate contains licensing parameters that dictate whether the merchandise is permitted to be executed. Thus, once properly licensed, the downloaded client components in conjunction with the electronic license certificate permit a legitimate customer to execute (process) purchased content in a manner that helps prevent illegitimate piracy.

In one embodiment, the electronic license certificate is generated from tables stored in a password generation data repository. Each table contains fields that are used to generate the license parameters. Each electronic license certificate is generated specifically for a particular item of merchandise and for a specific customer request. Also, the electronic license certificate is secured, such as by encryption, to prevent a user from accessing the corresponding item of merchandise without proper authorization. One technique for securing the electronic license certificate uses a symmetric cryptographic algorithm.

The secure digital commerce system also supports the ability to generate emergency electronic license certificates in cases where an electronic license certificate would not normally be authorized. To accomplish this objective, a separate emergency password generation table is provided by the password generation data repository. In addition, the secure digital commerce system reliably downloads the client components even when a failure is encountered during the download procedure. Further, a minimum number of components are downloaded.

In addition to generating electronic license certificates, the licensing and purchasing broker may also include access

to a payment processing function, which is invoked to authorize a particular method of payment for a particular transaction. The licensing and purchasing broker may also include access to a clearinghouse function used to track and audit purchases.

Digital commerce is performed using the secure digital commerce system as follows. A customer invokes an online purchasing system to request an item of merchandise and to indicate a purchasing option (such as "try" or "buy"). The DCS client then downloads onto a customer computer system the client components that are associated with the requested item. Included in these components is a secured content component. The secured content component is then installed and executed (processed) in a manner that automatically invokes licensing code. The licensing code, when the requested item is not yet licensed properly, causes the requested item to be licensed by the licensing and purchasing broker in accordance with the indicated purchasing option before the content component becomes operable. Specifically, the licensing and purchasing broker generates a secure electronic license certificate and completes an actual purchase when appropriate. The broker then returns the electronic license certificate to the licensing code, which unsecures (e.g., unencrypts) and deconstructs the electronic license certificate to determine the licensing parameters. The licensing code then executes (processes) the content component in accordance with the license parameters.

In some embodiments, the secure digital commerce system supports the licensing and purchasing of both merchandise that is deliverable online and merchandise that requires physical shipment of a product or service (e.g., non-ESD merchandise).

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates how a browser application, using the client/server model of the World Wide WEB network, enables users to navigate among network nodes by requesting and receiving WEB pages.

FIG. **2** is an example display screen of an online virtual store that operates with the secure digital commerce system.

FIG. **3** is an overview block diagram of the secure digital commerce system.

FIG. **4** is an overview flowchart of the example steps performed by the secure digital commerce system components to perform the licensing and purchase of electronic data.

FIG. **5** is a block diagram of a general purpose computer system for practicing embodiments of the DCS client.

FIG. **6** is an example flow diagram of the steps performed to generate the components of the DCS client.

FIG. **7** is an example WEB page of a virtual store used to purchase electronic data, which is executing on a customer computer system.

FIG. **8** is an example flow diagram of the steps performed by a boot program executed on a customer computer system to download client components when licensing a selected item of merchandise.

FIG. **9** is an example flow diagram of licensing code that has been incorporated into an encrypted content file.

FIG. **10** is an example display screen presented by a virtual store to determine whether a customer desires to license a product for trial use or for purchase.

FIG. **11** is an example flow diagram of the steps performed by licensing code to determine whether a valid electronic licensing certificate is available.

FIG. **12** is an example flow diagram of the steps performed by a licensing and purchasing broker of the secure digital commerce system.

FIG. **13** is an example display screen of the WinZip 6.2 program, which was selected for purchase in FIG. **7**, when it executes after completing the licensing procedures.

FIG. **14** is an example display screen for selecting a particular credit card.

FIG. **15** is an example display screen for entering a password for a selected credit card.

FIG. **16** is an example display screen for adding a new credit card.

FIG. **17** is an example display screen for allowing a customer to verify an intent to purchase after supplying a method of payment.

FIG. **18** is an example display screen for indicating that a purchasing transaction has been authorized.

FIG. **19** is an example block diagram that illustrates one technique for ensuring secure communications between a DCS client component and a licensing and purchasing broker.

FIG. **20** is an example encrypted message protocol for sending encrypted messages between a DCS client component and a licensing and purchasing broker.

FIG. **21** is an example flow diagram of the additional steps performed by a licensing and purchasing broker of the secure digital commerce system to support non-ESD transactions.

DETAILED DESCRIPTION OF THE INVENTION

Exemplary embodiments of the present invention provide methods and systems for facilitating secure digital commerce of electronic content. The secure digital commerce system interacts with an online purchasing system, such as a virtual store, to facilitate the purchase and distribution of merchandise over a network, such as the Internet or the World Wide WEB network (the WEB). For the purposes of this specification, a virtual store is any executable file, data, or document (for example, a WEB page) that enables a user to electronically purchase merchandise over a network.

FIG. **2** is an example display screen of an online virtual store that operates with the secure digital commerce system. Although the secure digital commerce system is described with reference to a virtual store, one skilled in the art will recognize that any type of electronic purchasing system or application, including a standalone application, is operable with embodiments of the present invention. A browser application window **201** is shown currently displaying (and executing) a WEB page **202** retrieved from the location specified by the URI "www.buysoftware.com." WEB page **202** provides a set of user interface elements, for example, pushbuttons **204** and **205** and icon **203** which display information or which can be used to navigate to additional information. A virtual store typically provides a set of icons, which each describe an item of merchandise that can be purchased. For example, graphical icon **203** is an example icon that is linked to the functionality needed to purchase a Microsoft Corp. software game entitled "RETURN OF ARCADE."

Each icon is typically linked to a server site on the network, which is responsible for supplying the content of the item when purchased if the item is capable of electronic delivery. When the user selects one of the icons, the browser application, as a result of processing the link, sends a request

for the selected item to the server site. Thus, when a customer selects the icon **203**, an HTTP request message is sent to an appropriate server site to locate and download the software modules that correspond to "RETURN OF ARCADE."

For the purposes of this specification, the merchandise that can be licensed and distributed online includes any type of digital or electronic, information or data that can be transmitted using any means for communicating and delivering such data over a network, including data transmitted by electronics, sound, laser, or other similar technique. Similarly, although the present application refers generically to "electronic data" or "electronic content," it will be understood that embodiments of the present invention can be utilized with any type of data that can be stored and transmitted over a network.

The secure digital commerce system is arranged according to a client/server architecture and provides a modularized DCS client and a modularized DCS server that interact with the online purchasing system to perform a purchase. The DCS client includes a set of client components; support for downloading the client components onto a customer computer system; and support for communicating with the DCS server to license an item of merchandise. The client components contain a secured (e.g., encrypted) copy of the content and various components needed to license and purchase the merchandise and to unsecure (e.g., decrypt) and execute the licensed merchandise. The DCS client communicates with the DCS server to download the client components onto a customer's computer system in response to a request for merchandise from the online purchasing system. The DCS client also communicates with the DCS server to license and purchase the requested merchandise. The DCS server generates an electronic license certificate, which contains license parameters (e.g., terms) that are specific to the requested merchandise and to a desired purchasing option (such as trial use, permanent purchase, or rental). The DCS server then sends the generated electronic license certificate to the DCS client. Once a valid electronic license certificate for the requested merchandise is received by the DCS client, the merchandise is made available to the customer for use in accordance with the license parameters contained in the electronic license certificate.

The DCS client includes a download file, a user interface library, a purchasing library, a secured content file, a DCS security information file, and licensing code. There is a download file for each item of merchandise that can be distributed electronically, which contains an executable boot program. The boot program is responsible for determining what components need to be downloaded for a requested item of merchandise. The secured content file contains the content that corresponds to the requested item of merchandise. The content may be a computer program, data, or a combination of both. For the purposes of this specification, "secure" or "secured" implies the use of cryptography or other types of security, including the use of hardware. One or more of the remaining components can be shared by several items of merchandise. For example, the user interface library, which defines a user interface used to purchase and license merchandise, may be specific to an item of merchandise or may be uniform for an entire online purchasing system. The purchasing library, licensing code, and DCS security information file are used to interact with the DCS server to properly license requested merchandise. In particular, the licensing code ensures that the requested merchandise is not operable by the customer until it has been properly licensed by the DCS server.

The DCS server includes a content supplier server, a licensing and purchasing broker, and a payment processing function. The content supplier server provides the merchandise-specific DCS client components. The licensing and purchasing broker generates electronic license certificates and manages purchases. The payment processing function authorizes payment for a particular transaction. One or more of each of these entities may be available in a DCS server.

One of the advantages of the modularized nature of exemplary embodiments of the present invention is that it provides a natural mechanism for replacing individual components and for customizing the system. For example, by replacing only the licensing code and a portion of the licensing and purchasing broker, an entirely new cryptographic algorithm may be used to secure the content. Embodiments of the invention also support the secure execution of requested merchandise and minimize the number of components needed to securely download, license, and execute the requested merchandise.

For the purposes of this specification, any client/server communication architecture and communication protocol that supports communication between the DCS client and the DCS server could be used.

However, in an exemplary embodiment, the secure digital commerce system utilizes the HTTP request communication model provided by the World Wide WEB network. A detailed description of this architecture and of WEB page communication is provided in J. O'Donnell et al., *Special Edition Using Microsoft Internet Explorer* 3, QUE Corp., 1996, which is incorporated herein by reference.

FIG. **3** is an overview block diagram of the secure digital commerce system. FIG. **3** includes a DCS client **301** and a DCS server **302**, which are used with an online purchasing application, such as a WEB browser application **303**, to provide a purchasing interface for a potential customer. The DCS client **301** includes a virtual store **304** and a data repository **305**. The virtual store **304** provides a customer front end **312** and stores in the data repository **305** merchandise-specific download files **313**. The customer front end **312** includes WEB pages and associated processing support, which are downloaded onto a customer computer system **311** to enable a user to purchase merchandise. The download files **313**, which each contain an executable boot program and a component list, are used to download the merchandise-specific client components (for example, a secured content file and licensing code). When an item of merchandise is requested, the associated download file is processed to extract the executable boot program and the component list. The executable boot program downloads the needed components from the content supplier server **306** using the component list, which specifies the components that are needed to successfully license and operate the corresponding item of merchandise. In an alternate embodiment, download files are generated dynamically from component lists, which lists are stored in the data repository **305**.

The DCS server **302** includes a content supplier server **306**, a licensing and purchasing broker (server) **307**, a password generation data repository **308**, and a payment processing function **309**. The licensing and purchasing broker **307** includes a separate licensing library **310** (passgen.dll), which contains the code for generating an appropriate license in response to a request from the virtual store. The licensing library **310** uses the password generation data repository **308** to generate an electronic license

certificate ("ELC") with licensing parameters that correspond to a particular item of merchandise. An electronic license certificate is encrypted electronic data that provides information that can be utilized to determine whether a particular customer is authorized to execute the merchandise. Such information may include, for example, the specification of a period of time that a particular customer is allowed to execute the merchandise for trial use. The data repository **308** contains tables and fields that are used to create the license parameters of a license. The data repository **308** may contain information that is supplied by the source companies of the available merchandise. The payment processing functions **309** are used by the licensing and purchasing broker **307** to charge the customer and to properly credit the appropriate supplier when the customer requests an actual purchase (rather than trial use or another form of licensing). In addition, clearinghouse functions may be invoked by the licensing and purchasing broker **307** to audit and track an online purchase. Clearinghouse functions may be as provided by well-known commercial sources, such as Litlenet and Cybersource. Similarly, payment processing functions may be provided using well-known commercial credit card authorization services.

FIG. **4** is an overview flowchart of the example steps performed by the secure digital commerce system components to perform the licensing and purchase of electronic data. This figure briefly describes the interactions between the components shown in FIG. **3** to accomplish the downloading, licensing, and purchasing of a requested item of merchandise when it can be delivered online. In step **401**, the potential customer downloads a WEB page (part of the customer front end **312**) from the virtual store **304** that includes the item to be requested (see, for example, FIG. **2**). In step **402**, the customer requests an item of merchandise, for example, by selecting an icon that is linked to a download file that corresponds to the desired item. In response to the selection, in step **403**, the virtual store **304** downloads and installs the download file, which extracts the executable boot program and component list and causes execution (preferably as a background task) of the executable boot program on the customer computer system **311**. In step **404**, the boot program reads the component list to determine what DCS client components to download and requests the determined components from the appropriate contents supplier server **306**. The component list, as further described below with reference to Table 2, indicates source and target locations for each component to be downloaded. In step **405**, the boot program installs a downloaded (secured) content file that is associated with the desired item of merchandise and causes the content file to be processed (executed). When the content file is a computer program, then the downloaded content file has been previously configured to automatically cause licensing code to be executed before the content file is executed. When instead the content file is data to be input to a computer program, then the content player is previously configured to automatically cause the licensing code to be executed first before the content file data is processed. More specifically, the downloaded content player is installed by the boot program to process the secured (e.g., encrypted) content file data. The boot program then starts the execution of the content player, which invokes and causes execution of the downloaded licensing code. Thus, in step **406**, the licensing code, which is incorporated into either the content file or the content player, is executed. In step **407**, if the licensing code determines that a valid ELC already exists, then the content file continues to be processed in step **412**, else the licensing code continues in step **408**. In step **408**, the licensing code requests a valid ELC from the licensing and purchasing broker **307**. In step **409**, the licensing and purchasing broker **307** determines whether a purchase is requested and, if so, continues in step **410**, else continues in step **411**. In step **410**, the licensing and purchasing broker **307** obtains a method for payment and authorizes the payment method using the payment processing function **309**. In step **411**, the licensing and purchasing broker **307** generates an appropriate ELC using the licensing library **310** and the password generation data repository **308** and returns the generated EL-C to the licensing code. In step **412**, if portions of the content file are encrypted as will be further described, then the content file is decrypted and processed.

As indicated above, when the downloaded (secured) content file is a computer program, licensing code is automatically invoked to verify the existence of, or obtain, a valid electronic license certificate for a requested item and to decrypt and execute the content file. One mechanism for incorporating licensing code into a content file such that it is automatically invoked is discussed in detail with reference to related U.S. patent application Ser. No. 08/792,719, entitled "Method and System for Injecting New Code Into Existing Application Code," filed on Jan. 29, 1997. That patent application describes a technique for inserting licensing code into an existing application and for inserted security code that securely executes the application code. The security code uses an incremental decryption process to ensure that a complete version of the unmodified application code is never visible at any one time (to avoid illegitimate copying). Thus the security code mechanism described therein makes it impossible for someone to create an unmodified version of the application in a reasonable amount of time. The insertion technique described therein can be used to insert into a content file the licensing code component of the DCS client, which communicates with the licensing and purchasing broker to generate an ELC. Further, the encryption/decryption technique described therein may be used in the current context to incorporate security code that securely decrypts and executes the downloaded content file.

In addition, when the content file is data to be used as input to a computer program (such as a content player), then the licensing code can be incorporated into the computer program by invoking licensing code and security code routines. For example, an application programming interface ("API") to the licensing code and to the incremental decryption security code can be provided. The content player is programmed (or configured via the insertion technique described in the related patent application) to include calls to the API routines to validate or obtain an ELC and to unsecure (e.g., decrypt) the associated content file. One skilled in the art will recognize that any mechanism that automatically causes the execution of licensing code (and security code) before the secured content is processed is operable with embodiments of the present invention.

In exemplary embodiments, the DCS client is implemented on a computer system comprising a central processing unit, a display, a memory, and other input/output devices. Exemplary embodiments of the DCS client are designed to operate in a globally networked environment, such as a computer system that is connected to the Internet. FIG. **5** is a block diagram of a general purpose computer system for practicing embodiments of the DCS client. The computer system **501** contains a central processing unit (CPU) **502**, a display **503**, a computer memory (memory) **505**, or other computer-readable memory medium, and other input/output devices **504**. Downloaded components of the DCS client

preferably reside in the memory **505** and execute on the CPU **502**. The components of the DCS client are shown after they have been downloaded and installed on the computer system **501** by an executable boot program and after an appropriate electronic license certificate has been generated and installed. Specifically, the components of the DCS client include the executable boot program **507** (SAFEboot); a user interface library **508** (SAFEUI.dll); a purchasing request library **509** (SAFEBuy.dll); an encrypted content file **510**, which is shown with incorporated licensing code **511** (SAFE.dll); an encrypted DCS security information file **512**, which is associated with the encrypted content file **510**; and an electronic licensing certificate **514** (ELC). As shown, each library is typically implemented as a dynamic link library (a "DLL"). In addition to these components, when the encrypted content file contains data that is not a computer program, the memory **505** contains a content player **513** for processing the content file **510**, which has incorporated licensing code **511**. Also, WEB browser application code **506** is shown residing in the memory **505**. Other programs **515** also reside in the memory **505**. One skilled in the art will recognize that exemplary DCS client components can also be implemented in a distributed environment where the various programs shown as currently residing in the memory **505** are instead distributed among several computer systems. For example, the encrypted content file **510** may reside on a different computer system than the boot program **507**.

In exemplary embodiments, the DCS server is implemented on one or more computer systems, each comprising a central processing unit, a memory and other input/output devices. Each of these computer systems may be a general purpose computer system, similar to that described in FIG. **5**, which is connected to a network. The server systems that comprise the server portion may or may not include displays. The password generation data repository may be implemented using any well-known technique for implementing a database or any other type of data repository. Although shown as a separate facility, one skilled in the art will recognize that the data repository may be incorporated as a component of the computer system that is used to implement the licensing and purchasing broker. Further, one skilled in the art will also recognize that a variety of architectures are possible and can be used to implement exemplary embodiments of the DCS server.

FIG. **6** is an example flow diagram of the steps performed to generate the components of the DCS client. In an exemplary embodiment, these steps are performed by a utility program referred to as the SAFEmaker utility. The SAFEmaker utility is responsible for generating the downloadable components that correspond to an item to be supplied as online merchandise. In addition, the utility generates a secured content file that can only be processed when access is granted. This capability is referred to as making the file "SAFE" (hence, the SAFE-prefix in the component names). Making a content file "SAFE" implies that security code and licensing code are incorporated into the content file (or content player, in the case of digital content that is not a computer program) to ensure that the online merchandise is usable only when proper licensing has been performed. Typically, this process involves encrypting some portion of the content file. Some components generated by the SAFEmaker utility are stored on the content supplier server (e.g., content supplier server **306** in FIG. **3**) and are downloaded in response to requests from the virtual store front end. Other components are stored on the virtual store, which may be located on a different computer system from the content supplier server. The SAFEmaker utility also updates the password generation data repository of the DCS server with merchandise-specific information.

Specifically, in step **601**, the utility incorporates licensing and security code into the supplier specific electronic content or content player. As described above, an exemplary embodiment incorporates licensing and security code according to the techniques described in the related U.S. patent application Ser. No. 08/792,719, entitled "Method and System for Injecting New Code into Existing Application Code," filed on Jan. 29, 1997 or by calling routines of an API as appropriate (e.g., when a content player is needed). One skilled in the art, however, will recognize that any technique for ensuring that proper licensing code gets executed when the content is processed and for encrypting (and subsequently decrypting) the content file will operate with embodiments of the present invention. In step **602**, the utility produces one or more files that contain the (partially or fully) encrypted content. In step **603**, the utility produces an encrypted DCS security information file(s), which contain information that is used, for example, to decrypt the content and to produce a proper license. The contents of an encrypted DCS security information file are described in further detail below with reference to Table 1. In step **604**, the utility creates a component list file (an ".ssc" file) and a download file for this particular online merchandise. Specifically, in an embodiment that statically generates download files, a self-extracting installation file is generated (the download file), which contains the component list file (an ".ssc" file) specific to the merchandise and the executable boot program. As described above, the download file, which contains the executable boot program and the component list, is typically stored on the virtual store computer system. The executable boot program uses the component list file to determine the components to download and to download them when particular electronic content is requested. An example component list file is described further below with reference to Table 2. In step **605**, the utility stores the download file on the virtual store computer system (e.g., virtual store **304** in FIG. **3**). When instead the download files are dynamically generated by the virtual store when needed for a particular WEB page, then in steps **604** and **605**, the utility creates and stores only the component list file. In step **606**, the utility stores the other components of the DCS client, for example, the encrypted content and DCS security information files, the licensing code, and the user interface library on the content supplier server system (e.g., content supplier server **306** in FIG. **3**). In step **607**, the utility updates the password generation data repository (e.g., password generation database **308** in FIG. **3**) with the merchandise-specific licensing information, for example, the fields used to generate the license parameters of a valid electronic license certificate, and then returns. An example password generation data repository is discussed in further detail with reference to Tables 3, 4, and 5. One skilled in the art will recognize that the generation of these components and the password generation data may be performed at different times and by separate utilities.

TABLE 1

Field Name:	Type:
CommerceServer	String
ProductSkuId	String
ProductUUID	String
UILibName	String

TABLE 1-continued

Field Name:	Type:
EntryPoint	Integer
ImageBase	Integer
Ekey	String
Ecode	BinaryObject
DataSize	Integer
NumberRelocations	Integer
Relocations	String
ContactCompany	String
ContactAddress	String
ContactSupportPhone	String
ContactSupportFax	String
ContactSupportEmail	String
ContactOrderPhone	String
ContactOrderFax	String
ContactOrderEmail	String
ProductName	String
LicenseFilename	String
LicenseAdminDir	String
DeveloperId	String
SecretKey	BinaryObject
ActiveAssistants	Integer
FeatureName	String
FeatureNumber	Integer
HostIdTypeList	String
IntegrationType	Integer

Table 1 is an example list of fields that may be included in an encrypted DCS security information file. For each encrypted content file (or set of files), the supplier provides fields that are used by a virtual store to download, license, and purchase the associated electronic content. The data in the encrypted DCS security information file is encrypted separately from the content file to enable multiple items of merchandise to share purchasing, licensing, and decryption information. This capability is especially useful when the items are provided by the same content supplier server. Thus, a single encrypted DCS security information file may be associated with more than one encrypted content file. In addition, each field in the DCS security information file is encrypted separately. By separately encrypting each field, purchasing or licensing information can be changed without having to re-encrypt the content file or the rest of the DCS security information file.

Specifically, in Table 1 the CommerceServer field indicates the location of the licensing and purchasing broker (e.g., the network address of licensing and purchasing broker **307** in FIG. **3**) to be used to license and purchase the merchandise. (In embodiments of the secure digital commerce system, one or more content suppliers, licensing and purchasing brokers, or payment processing functions, may be utilized.) The ProductSKUId field is a specific identifier associated with a version (each executable) of a product for a specific reseller (virtual store). For the purposes of example, exemplary embodiments assume that a product may have multiple versions and that each version may be packaged differently depending upon the purchasing option (for example, trial use versus full purchase). In addition, more than one reseller may offer a version of a product. The ProductSKUId field is used to identify a password configuration table to be used to generate an electronic license certificate and is discussed further below. The ProductUUID field is a specific identifier associated with each version of a product regardless of the reseller. By using an identifier that is specific to the product version and not to the reseller, the digital commerce system can ensure that a customer who licenses a version of a product for (one time) trial use may not utilize multiple resellers to obtain more than one ELC for the same version. In addition, this identifier is used by the licensing code to locate the associated DCS security information file and is associated with various licensing-specific information. For example, clock data can be stored in a system registry indexed by ProductUUID to ensure that "time-bomb" protected content is not defeated by resetting the clock to illegitimately process the content. The UILibName indicates the location of a user interface library to be used for purchasing the merchandise. The EntryPoint, ImageBase, EKey, ECode, DataSize, NumberRelocations, and Relocations fields are used to support the decryption of the encrypted content file(s) and to determine the relocation information when the content file is secured using the technology of related U.S. patent application Ser. No. 08/792,719. If an alternative licensing and encryption scheme is used, then these fields would be modified accordingly. The ContactCompany, ContactAddress, ContactSupportPhone, ContactSupportFax, ContactSupportEmail, ContactOrderPhone, ContactOrderFax, and ContactOrderEmail fields reflect supplier dependent information that can be displayed in dialogs presented by the virtual store depending on the user interface being employed. The DeveloperID and SecretKey fields are used to create a symmetric key to decode the electronic license certificate generated by the licensing and purchasing broker. The other fields are used for other similar licensing and purchasing functions.

TABLE 2

```
<Execute
    TRIGGER     =   "<ProgramFilesDir>\winzip\winzip32.exe"
    URI         =   "http://devserver/products/winzip32/winzipsetup.exe"
    MSGDIG      =   "NDLsrKcS36YbugITP4yUjv8PSfk="
    ProductUUID =   "WINZIP-demo-0000"
    NAME        =   "WinZip 6.2"
    DESCRIPTION =   "WinZip 6.2"
    LOCAL       =   "<ProgramFilesDir>\winzip\setup.exe">
```

Table 2 is an example of the contents of a single entry in a component list file. In an exemplary embodiment, each icon in the virtual store that corresponds to an item that can be purchased and distributed online is associated with a component list file (an .ssc file). Within each component list file there is an entry similar to that shown in Table 2 for each component that is to be downloaded when the associated item is requested. For example, if there is an item-specific encrypted DCS security information file and an item-specific user interface library that are to be downloaded to purchase the requested item, then there are entries for each such component.

Each entry contains a tag that specifies how to process the component when it is downloaded and sufficient information to download a component if the file indicated by the TRIGGER field is not already present on the customer computer

system. Specifically, the tag (in this example "Execute") specifies what to do with the component referred to by the LOCAL field once it is downloaded. An "Execute" tag specifies that the component referred to by the LOCAL field (e.g., "setup.exe") will always be executed. A "Component" tag specifies that the component referred to by the LOCAL field is to be downloaded with no further processing. An "ExecuteOnce" tag specifies that the component referred to by the LOCAL field is to be executed only if the file referred to by the TRIGGER field does not already exist. The TRIGGER field of each entry indicates the location of a file that is present when the component does not need to be downloaded. Thus, the TRIGGER field is used to determine whether to download a component. The URI field indicates the location of a content supplier server that can provide the component. In addition, the MSGDIG field contains a message digest, which is used to determine whether the component has been successfully loaded. Use of the message digest is described in further detail below with respect to FIG. **8**. The ProductUUID, NAME, and DESCRIPTION fields indicate identifying information used by the licensing code. When present, these fields are typically stored in a system registry and used by the licensing code to determine which DCS security information file to use for a particular content file. In addition, the NAME field may be displayed by the boot program executable to give user feedback regarding the component currently being downloaded. The LOCAL field indicates a target location for the downloaded component on the customer computer system.

FIGS. **7–13** describe in further detail the steps performed by the secure digital commerce system to perform the licensing and purchasing process presented in FIG. **4**. One skilled in the art will recognize that these steps can be performed in other orders and by different components than those presented herein. As a preliminary matter, the customer first navigates to a virtual store WEB page in order to request an item for purchase. FIG. **7** is an example WEB page of a virtual store used to purchase electronic data, which is executing on a customer computer system. (Display of this WEB page corresponds to step **401** in FIG. **4**.) WEB page **701** contains an icon **702**, which, when selected, causes the "WinZip 6.2" product to be licensed and optionally purchased. Text area **703** contains descriptive text to aid a customer in making a decision to license or buy the WinZip 6.2 product. Pushbuttons **704** enable the user to explore other merchandise available for license and purchasing.

When the customer requests an item of merchandise to be licensed or purchased (for example, when the user selects icon **702** in FIG. **7**), then the virtual store downloads and potentially initiates the execution of a boot program associated with the requested merchandise (see step **403** in FIG. **4**). Specifically, each merchandise icon is linked (anchored) to a merchandise-specific download file, which is a file stored on (or generated by) the virtual store. In one embodiment, the download file is a self-extracting file that contains: extraction code, a header that indicates the size of the boot program which follows, the boot program (preferably compressed), and the appropriate component list file. The download file can be generated statically using the SAFEmaker utility described above or can be generated dynamically by the virtual store when it downloads a WEB page that includes the icon that is anchored to the download file. When the customer selects a merchandise icon, the customer is queried whether to download and store or download and execute the anchor file (indicated by the link). When the user indicates that the download file is to be executed, the extraction code of the download file is executed, which causes the component list (the ".ssc" file) to be extracted and the boot program executable to be (potentially decompressed,) extracted and executed. One skilled in the art will recognize that any mechanism for associating an icon with a boot program and for causing the boot program to be downloaded and executed is operable with the secure digital commerce system.

FIG. **8** is an example flow diagram of the steps performed by a boot program executed on a customer computer system to download client components when licensing a selected item of merchandise. (These steps correspond to steps **404–405** in FIG. **4**.) The boot program is implemented such that it downloads only the components that are necessary to license (and optionally purchase) the selected item. For example, if the user interface library to be used to purchase the selected item is the same library as one already downloaded, then it is not downloaded again. In addition, the boot program can recover from a failure during the load process and can resume downloading where it left off. The boot program accomplishes these objectives by using a message digest algorithm to determine whether a component has been successfully downloaded onto a customer computer system.

Specifically, in step **801**, the boot program reads the component list (the ".ssc" file) associated with the selected item of merchandise to determine what components to download from a specified content supplier server. In steps **802–808**, the boot program executes a loop to process each remaining component in the component list that has not already been successfully downloaded. Specifically, in step **802**, the boot program selects the next component from the component list that appears following the last successfully read component. In step **803**, the boot program determines whether all of the remaining components of the list have been processed, and if so, returns, else continues in step **804**. In step **804**, the boot program determines whether the file indicated by the TRIGGER field is already present. If not, the boot program obtains the component indicated by the URI value from the content supplier server and stores the obtained component as indicated by the LOCAL value (see Table 2). In step **805**, the boot program calculates a message digest (the value of a one-way hash function) for the downloaded component. In step **806**, the determined message digest for the newly downloaded component is compared with a previously stored message digest in the component list (see the MSGDIG value in Table 2). In an exemplary embodiment, an MD**5** algorithm is used to calculate a message digest. However, one skilled in the art will recognize that any message digest algorithm or any function capable of determining a predictable value for the downloaded component for comparison to an already stored value may be used. The MD**4** and MD**5** algorithms are described in Bruce Schneier, *Applied Cryptography*, John Wiley & Sons, Inc., 1994, which is hereby incorporated by reference. In step **807**, if the calculated message digest is identical to the stored message digest, then the boot program continues in step **808**, else continues back to the beginning of the loop in step **802**, because a failure has occurred in downloading the component. In step **808**, the boot program sets an indicator of the last successfully read component to indicate the component most recently loaded. In step **809**, the boot program processes the component according to the tag (e.g., "Execute"), and continues back to step **802** to select the next component to download. Note that the tag associated with each component entry will automatically cause the secured content file (or the content player, depending on the situation) to begin executing.

One skilled in the art will recognize that different behaviors will occur when the content file (or content player) begins executing depending upon the technique used to incorporate the licensing code and decryption (security) code and depending upon the encryption/decryption technique used. For example, as described in further detail in related U.S. patent application Ser. No. 08/792,719, when using the injection techniques described therein, the execution of the encrypted content file will automatically cause the licensing code and (eventually) the security code to be executed as a result of injecting a licensing DLL into the content file. Specifically, a "DLLMain" routine is automatically invoked when the licensing code library is loaded, which in turn executes the actual licensing code. After the licensing code executes, the security code stored in the encrypted content automatically executes because it is inserted into the content file immediately following (a reference to) the licensing code. Thus, the licensing code and the decryption code are automatically executed before any supplier-specific content is executed. The security code in an exemplary embodiment decrypts the encrypted content incrementally in order to prevent a fully decrypted version of the content to be present in its entirety at any one time. A similar procedure is used when the content player invokes the licensing and security code with an exception that the licensing and security code is explicitly invoked and knows how to locate the content file and to decrypt it incrementally.

FIG. **9** is an example flow diagram of licensing code that has been incorporated into an encrypted content file. Similar code is incorporated in a content player by calling appropriate routines. The licensing code will be discussed for purposes of example relative to an encrypted content file. In one exemplary embodiment, the licensing code is provided in a dynamic link library, such as SAFE.dll **511** in FIG. **5**. (The steps of FIG. **9** correspond to steps **406–408** and **412** in FIG. **4**.) Each time the encrypted content file is executed by the customer computer system, the licensing code is preferably automatically executed. The licensing code is responsible for determining whether a valid electronic license certificate is available and, if so, allowing execution of the content, otherwise forcing the customer to license the item from the supplier.

Specifically, in step **901**, the licensing code determines whether a valid electronic license certificate ("ELC") is available. The steps used to make this determination are discussed further below with reference to FIG. **11**. If a valid ELC is available, then the licensing code continues in step **909** and skips the licensing and purchasing process, else continues in step **902**. In step **902**, the licensing code loads the user interface library associated with the component and obtains a purchase option from the customer, such as "rent-to-buy," "buy," or "try." The purchase options assist in determining the parameters of a valid license. An example interface for obtaining this information is described below with reference to FIG. **10**. The licensing code obtains the user interface library name by retrieving the UILibName field from the DCS security information file associated with the product. The associated DCS security information file can be determined from the ProductUUID, which was previously stored in the system registry by the boot program during the component download process. In step **903**, the licensing code determines whether the customer has indicated that a trial purchasing option is requested and, if so, continues in step **904**, else continues in step **905**. In step **904**, the licensing code sends an HTTP request message to the licensing and purchasing broker (e.g., the licensing and purchasing broker **307** in FIG. **3**) to provide an appropriate license for trial use of the product, and continues in step **908**. In step **905**, the licensing code determines whether the customer has indicated a purchasing option to purchase the content and, if so, continues in step **906**, else continues in step **907**. In step **906**, the licensing code sends an HTTP request message to the licensing and purchasing broker to purchase the content, and continues in step **908**. In step **907**, the licensing code determines whether any other type of licensing or purchasing request has been indicated by the customer and sends an appropriate HTTP request message to the licensing and purchasing broker. For example, other requests associated with rental use or other types of purchasing options may be supported. The processing of these HTTP request messages by the licensing and purchasing broker is discussed further below with respect to FIG. **12**. In step **908**, the licensing code receives a valid ELC from the licensing and purchasing broker, stores it, and continues in step **909**. The ELC may be stored in any area that is accessible to processes executing on the customer computer system, such as in a system registry. In step **909**, the licensing code causes the decryption and execution of the licensed content, and returns.

In an exemplary embodiment, the licensing code uses an intermediary library function (stored in, for example, the SAFEBuy.dll **509** in FIG. **5**) to send the purchasing request of step **906** to the licensing and purchasing broker. A separate library is useful in scenarios where other types of programs (other than virtual stores) desire to utilize the purchasing capabilities of the licensing and purchasing broker. The library function provides a unique transaction identifier that can be used to identify the particular purchase transaction at a further time. Such capability is useful, for example, to later cancel the purchase. One skilled in the art will recognize that other organizations of the licensing and purchasing support code are also possible.

FIG. **10** is an example display screen presented by a virtual store to determine whether a customer desires to license a product for trial use or for purchase. This display screen **1001** may be used to implement step **902** in FIG. **9**. When the customer selects the "Try" pushbutton **1002** in FIG. **10**, then the customer has indicated that trial use of the product is desired. Alternatively, when the customer selects the "Buy" pushbutton **1003** in FIG. **10**, then the customer has indicated the desire to purchase the product.

FIG. **11** is an example flow diagram of the steps performed by licensing code to determine whether a valid electronic licensing certificate is available. In step **1101**, the code retrieves, decrypts, and decodes the electronic licensing certificate (ELC) to obtain the parameters of the license (e.g., the license terms). The license parameters that are obtained in step **1101** indicate, for example, how many uses of a particular license can be executed or, for example, how many different user passwords are able to use the same electronic license. In addition, license parameters that reflect an authorized time period for use may be specified. In step **1102**, the code tests various attributes of the customer computer system to determine whether the conditions indicated by the retrieved license parameters have been met. In step **1103**, if all of the conditions have been met (for example, the license use period has not expired), then the code returns indicating that a valid license is in effect. Otherwise, the code returns indicating that the current license is invalid.

In an exemplary embodiment, the ELC is encrypted and decrypted using a symmetric key algorithm. A symmetric algorithm implies that the same key is used to encrypt a plaintext message and to decrypt a ciphertext message. Any

symmetric key algorithm could be used. Symmetric and public key cryptography, both of which are utilized by exemplary embodiments of the present invention, are described in detail in Bruce Schneier, *Applied Cryptography*, John Wiley & Sons, Inc., 1994, which is herein incorporated by reference. According to one technique, the DeveloperID and SecretKey fields (stored in the encrypted information file) are used to formulate a symmetric key, which is client and product specific. These fields are provided by the supplier when the SAFEmaker utility is executed to produce the components of the DCS client (see FIG. **6**). Because the encryption of the ELC is provided by the licensing and purchasing broker and the corresponding decryption of the ELC is provided by the licensing code, the encryption and decryption code are preferably synchronized to correspond to one another. For this reason, a separate dynamic link library (e.g., passgen.dll) is used by the licensing and purchasing broker to allow the encryption algorithm to be replaced at any time to correspond to different licensing code.

FIG. **12** is an example flow diagram of the steps performed by a licensing and purchasing broker of the secure digital commerce system. These steps are executed in response to receiving an HTTP request message sent by the licensing code in step **904** or **906** in FIG. **9**. As described earlier, the licensing and purchasing broker interacts with a password generation system (e.g., passgen.dll and the data repository) and payment processing functions to license and purchase an indicated item of merchandise. In summary, when the licensing and purchasing broker receives a request to buy an item, it performs appropriate payment processing to perform a purchase. When the licensing and purchasing broker receives either a request to try or a request to buy the item, the broker uses the password generation system to generate an ELC to return to the licensing code.

Specifically, in step **1201**, the broker determines whether a buy request has been received and, if so, continues in step **1202**, else continues in step **1206**. In step **1202**, the broker causes the licensing code (specifically, the user interface library routines) executing on the customer computer system to obtain credit card or purchase order information if such information was not already sent with the request. A sample user interface for obtaining method of payment information and for verifying the purchase transaction are described below with reference to FIGS. **14–17**. Once the credit card or purchase order information has been obtained by the licensing and purchasing broker, then in step **1203** the broker obtains payment authorization from a payment processor such as the payment processing function **309** in FIG. **3** and informs the licensing code accordingly. One skilled in the art will recognize that any mechanism for is authorizing use of a credit card could be used. In step **1204**, the customer's credit card account is charged, and the supplier system is automatically credited. One skilled in the art will recognize that the licensing and purchasing broker can either credit the supplier directly at this time by sending the appropriate information to the credit card company, or can have the credit card company pay the licensing and purchasing broker, which in turn is responsible for payment to the supplier. In step **1205**, the broker informs the licensing code of payment authorization and continues in step **1207**. An example user interface for reporting the transaction identification information to the customer is described below with reference to FIG. **18**. If payment has not been authorized, then the broker returns such information to the licensing code, discontinues execution of the steps in FIG. **12**, and fails to generate a valid ELC.

In step **1206**, the broker determines whether it has received an HTTP request message that indicates trial use is desired and, if so, continues in step **1207**, else continues in step **1209**. In step **1207**, in order for the broker to generate an ELC specific to the user and to the indicated product, certain information is typically sent by the licensing code in the HTTP request message. Specifically, information that uniquely identifies the user and the product version is provided. The broker uses the received product version identifier (the ProductSKUId) to retrieve from a version table a corresponding password configuration identifier (pass-config-id). Once the pass-config-id is retrieved from the version password generation data repository table, this identifier is used as an index into a password configuration table to determine a set of fields to be used to generate the license parameters of the ELC. (One will recall that the fields stored in the password generation tables were specified by the supplier of the content in conjunction with the SAFEmaker utility.) An example password configuration table is shown below as Table 3. A table with potentially different fields exists for each unique pass-config-id. Because multiple versions of products and multiple products may use the same pass-config-id, they may share a single password configuration table. This attribute may be useful, for example, if all the products from a particular supplier have similar electronic licensing capabilities. In step **1208**, an ELC is generated based upon the fields of the determined password configuration table using a symmetric key formulated from the SecretKey and DeveloperID fields of the encrypted information file and an appropriate encryption algorithm, as discussed earlier. For the purposes of this specification, the ELC may be viewed as a very long number which encrypts the license parameters indicated by the fields in the password configuration table. In an exemplary embodiment, the code used to perform steps **1207–1208** is provided in a separate code module (e.g., passgen.dll) so that the password generation code, including the encryption and decryption algorithms, can be easily replaced in a licensing and purchasing broker.

In step **1209**, the broker processes any other type of purchasing option, for example, a renting option, and generates an appropriate ELC in a similar fashion to steps **1207–1209**. In step **1210**, the broker sends the generated ELC back to the licensing code executing on the customer computer system, and then returns.

Once the licensing and purchasing broker has completed its generation and return of a valid electronic license certificate, the requested merchandise is then processed as described in step **412** of FIG. **4**. FIG. **13** is an example display screen of the WinZip **6.2** program, which was selected for purchase in FIG. **7**, when it executes after completing the licensing procedures.

FIGS. **14–17** provide sample user interface display screens that are displayed by the licensing code (via the user interface library) to retrieve method of payment information. These display screens may be presented in response to requests from the licensing and purchasing broker for more information. The particular display screens presented are determined by the user interface library that is associated with the downloaded content file or by a default user interface available for the virtual store (see e.g., SAFEUI.dll **508** in FIG. **5**). As mentioned, the appropriate user interface library is determined by the licensing code from the UILibName field of the DCS security information file. FIG. **14** is an example display screen for selecting a particular credit card. FIG. **15** is an example display screen for entering a password for a selected credit card. The credit card data is

sent to the licensing and purchasing broker in encrypted form. In an exemplary embodiment, the credit card information is stored on the customer computer system using a secure technique. One such technique is known as "wallet technology." Wallet technology is an ActiveX control supplied by Microsoft Corp., which encrypts credit card information on a client's hard disk and keeps track of all credit cards. FIG. **16** is an example display screen for adding a new credit card. FIG. **17** is an example display screen for allowing a customer to verify an intent to purchase after supplying a method of payment. The display screen includes pricing information, which is supplied to the licensing code by the licensing and purchasing broker using the password generation data repository. Once the user has selected the Buy pushbutton **1702** in FIG. **17** indicating agreement to purchase the merchandise at the displayed price, the credit card (or purchase order) information is forwarded to the licensing and purchasing broker. FIG. **18** is an example display screen for indicating that a purchasing transaction has been authorized by the licensing and purchasing broker and the particular transaction identifier.

Communications between the DCS client components and the licensing and purchasing broker are preferably performed using a secure communication methodology. FIG. **19** is an example block diagram that illustrates one technique for ensuring secure communication between a DCS client component and a licensing and purchasing broker. Although FIG. **3** may imply that the downloaded components communicate with the licensing and purchasing broker to request licensing and purchasing and to receive the generated ELC, one skilled in the art will recognize that it is also possible for these components to communicate via a server associated with the virtual store. In FIG. **19**, communication between the client components (clients) **1901** and **1902** and the licensing and purchasing broker **1903** depends upon secure key exchange. Secure key exchange is accomplished by sending a client-specific symmetric key using a public/private key algorithm. The client-specific symmetric key is used solely for communication between that client and the licensing and purchasing broker. Specifically, a separate communication session-specific symmetric key is provided by each client for each communication session and is sent to the licensing and purchasing broker **1903** in a session initiation message using the broker's public key. One technique for distributing and obtaining the broker's public key is to use a commercially available digital signature service, such as Verisign. Because the broker **1903** is the only process that knows its own private key, the broker **1903** decrypts the session initiation message using its private key and retrieves the client's session-specific symmetric key. Thereafter, all messages from the broker **1903** to the client **1901** are encrypted by the broker **1903** using the client **1901**'s symmetric key. Client **1901** is then able to decrypt a received message using the symmetric key that it initially generated and sent to the broker **1903**. Client **1901** encrypts messages to send to the broker **1903** also using client **1901**'s symmetric key. Similarly, the client **1902** sends its own encrypted symmetric key to broker **1903** using the broker's public key. The broker **1903** in turn communicates with the client **1902** using the client-specific symmetric key that corresponds to client **1902**.

One skilled in the art will recognize that any algorithm for generating a symmetric key may be utilized. One skilled in the art will also recognize that any symmetric cryptographic algorithm that utilizes a symmetric key may be used to encrypt and decrypt the messages. For example, the DES algorithm, which is described in detail in the Schneier reference, could be utilized. In an exemplary embodiment, the RC**5** algorithm, which is a proprietary symmetric key algorithm available from RSA Data Security, Inc., is utilized. In addition, any cryptographic algorithm that uses public/private pairs of keys may be utilized to implement the technique described with reference to FIG. **19**. In an exemplary embodiment, the public/private key pairs are generating according to the RSA public-key algorithm. This algorithm is described in further detail in the Schneier reference.

FIG. **20** is an example encrypted message data structure for sending encrypted messages between a DCS client component and a licensing and purchasing broker. Plaintext message **2001** is encrypted as specified in FIG. **19** and stored according to the layout of ciphertext message **2002**. Ciphertext message **2002** contains a message digest **2003** and an encrypted symmetric key **2004**, which has been encrypted using the licensing and purchasing broker's public key. In addition, field **2005** contains the message content, which has been encrypted using the symmetric key that is sent in encrypted form in field **2004**.

Tables 3–5 are example password generation tables stored in the password generation data repository, which is used by the licensing and purchasing broker to generate electronic license certificates.

TABLE 3

Password-Configuration Table

Field	Type
pass-config-id	Varchar
password-version	Int
secret-key	Varchar
developer-id	Varchar
expire-password-in	Varchar
start-date	Varchar
password-output-scheme	Int
developer-info	Varchar
concurrent-code	Int
Licenses	Int
soft-licenses	Int
program-executions	Int
flex-nodelock-machines	Int
maximum-usernames	Int
release-number	Int
minor-release-number	Int
hostid-type	Int
misc-info	Int
min-hostids	Int
max-hostids	Int
instances	Int
emergency-id	Varchar
feature-type	Int
feature-list	Varchar

Table 3 is an example password configuration table. As described earlier, a separate password configuration table is provided for each password configuration identifier (pass-config-id). There is a version table in the data repository for translating between a retailer specific product version identifier (the ProductSKUId) and a corresponding password configuration identifier. The fields are used to generate the license parameters for an ELC that corresponds to the determined password configuration identifier. One skilled in the art will recognize that any fields could be stored in the password configuration table. Further, any algorithm for combining the fields in a determinable fashion to encrypt them into a single code that can be decrypted without losing information could be utilized to generate the ELC.

TABLE 4

Generated-Passwords Table

Field	Type
pass-config-id	Varchar
user-id	Varchar
generation-type	Int
date-generated	datetime
password	Varchar

Table 4 is an example table of the actual passwords generated for a particular password configuration identifier (pass-config-id). One of these tables exists for each password configuration identifier. Further, both normal passwords and emergency passwords (discussed below) are stored in this table. User identification information is also included for each generated password.

TABLE 5

Emergency-Password Table

Field	Type
emergency-id	Varchar
user-id	Varchar
pass-config-id	Varchar
start-hour	Int
end-hour	Int
start-minute	Int
end-minute	Int
start-day-number	Int
end-day-number	Int
start-date	Int
end-date	Int
start-month	Int
end-month	Int
start-year	Int
end-year	Int
start-week-number	Int
end-week number	Int

Table 5 is an example emergency password table. An emergency password table is used by the licensing and purchasing broker to generate an emergency password when a customer has for some reason lost a valid ELC (and potentially the merchandise), but has been previously authorized to use the merchandise. Emergency passwords are particularly useful in a scenario where the customer is unable to reach the supplier of the merchandise using available contact information. For example, if the customer's hard disk is destroyed during a weekend, it is useful to be able to re-generate a valid ELC and potentially re-download the merchandise to allow the customer to continue to utilize an already purchased product.

More specifically, the virtual store supports the creation of software on a removable medium, such as a floppy disk, which can be used to recreate the merchandise. When the customer's system hard disk fails, as part of recreating the system, the customer runs a merchandise recovery program from the removable disk. The recovery program has previously stored the boot programs and the component lists associated with the merchandise already purchased so that the relevant files can be resurrected. In addition, the recovery program attempts to create a new ELC using the normal password configuration table (e.g., Table 1). However, if the fields stored in the normal password configuration table do not allow for the creation of a new ELC for that user (for example, the number of uses remaining=0), then an emergency, temporary password is generated. The fields shown in Table 5 are used to generate the emergency ELC when the normal password generation table will not allow for the generation of an additional ELC. In that case, an ELC is generated that expires within a certain amount of time, for example 24 hours, to ensure that the customer calls the supplier's customer service number as soon as possible. The fields of the emergency password table are combined to generate an (encrypted) ELC in the same manner described with reference to Table 3. Emergency passwords once generated are also stored in entries in the generated password table, Table 4.

The description thus far has primarily referred to use of the components of the client portion of the secure digital commerce system by a virtual store. One skilled in the art will recognize that many alternative configurations are possible. For example, a standalone online purchasing application can be used to execute the components of the DCS client to communicate directly to a licensing and purchasing broker to request and receive electronic licensing certificates. In addition, one skilled in the art will recognize that the separate components of the DCS client and the DCS server enable each component to be separately replaceable and separately customized. For example, to generate a customized virtual store, a specialized user interface for licensing and purchasing merchandise can be generated and stored as the user interface component (e.g., SAFEUI.dll **508** in FIG. **5**) on the customer computer system. Further, one skilled in the art will recognize that the licensing code incorporated into the encrypted content (or content player) can be replaced in its entirety and can be made supplier specific. In addition, the code used to generate ELCs from the password generation data repository can be optimized to be supplier specific. Further, all of the functions of the DCS server can be provided as licensing and purchasing administrative functions (for example, via an applications programming interface) to enable content suppliers to furnish their own licensing and purchasing brokers.

The secure digital commerce system can also be utilized to support a combination of transactions pertaining to the online delivery of goods with transactions pertaining to physically deliverable goods and services. For example, along with the purchase of the WinZip 6.2 computer program, the virtual store may offer merchandise, such as mugs, T-shirts, travel bags, and even support service packages that cannot be delivered online. In these instances, the licensing and purchasing broker is additionally responsible for classifying received requests into online deliverables (ESD items) and into physical deliverables (non-ESD items) and is responsible for ordering and purchasing the non-ESD items.

FIG. **21** is an example flow diagram of the additional steps performed by a licensing and purchasing broker of the secure digital commerce system to support non-ESD transactions. In step **2102**, the licensing and purchasing broker selects the next item of merchandise requested starting with the first. FIG. **21** assumes that each HTTP request may request more than one item of merchandise. For example, a user interface library may offer additional non-ESD merchandise, which can be purchased at the same time that a customer purchases an ESD item. The user interface library generates and sends to the licensing and purchasing broker an HTTP request, which requests the purchase of multiple items of merchandise. For each item in the purchase request, in steps **2103–2110**, the broker processes the item in accordance with an indicated purchasing option for the item.

Specifically, in step **2102**, the broker determines whether there are more items remaining to be processed for the

request and, if so, continues in step **2103**, else finishes processing. In step **2103**, the licensing and purchasing broker determines whether the item is an ESD item or a non-ESD item. One mechanism used to determine whether the item is an ESD or a non-ESD item is to store a flag in the version table in the password generation data repository. For each purchasable item (ProductSkuId), the version table stores either a password configuration identifier or a distributor information identifier. In step **2104**, if the item is an ESD item, then the broker continues in step **2105**, else continues in step **2106**. In step **2105**, the broker executes the steps previously discussed with reference to FIG. **12** for items that are deliverable online. In step **2106**, the broker determines distributor contact information for the non-ESD item from a distributor information table stored within a data repository. The distributor information table for non-ESD transactions can be stored along with the password generation tables in the password generation data repository or in its own data repository. The distributor information stored in the table includes sufficient location information for contacting a distributor from whom the item can be purchased using an electronic request. In step **2107**, the broker obtains preauthorization information for a method of payment specified by the customer. It is assumed in this step that such information has been already obtained. If necessary, however, the broker sends appropriate requests to the code that initiated the purchase request (for example, the user interface library) to obtain method of payment information from the user and to continue accordingly. Preauthorization is necessitated by non-ESD purchases, which require a shipment date before the broker is able to charge the purchase to a customer's credit card. The preauthorization is performed by the payment processing function (e.g., the payment processing function **309** in FIG. **3**). In step **2108**, if the purchase is preauthorized, then the broker continues in step **2109**, else continues in step **2110**. In step **2109**, the broker sends a purchase order to the located distributor for the merchandise using a well-known Electronic Data Interchange ("EDI") format and commercial EDI products, such as those provided by Digital Corporation. One skilled in the art will recognize that any mechanism that allows information for electronically providing a purchase order would be operable with the licensing and purchasing broker. In step **2110**, the broker returns the results of the preauthorization attempt to the requesting routine, and then returns to the beginning of the loop in step **2101**.

To complete the purchasing transaction for a non-ESD item, the licensing and purchasing broker waits until it is informed by the distributor that the distributor will fulfill the requested purchase order (ship the merchandise) on a particular date. At that time, the licensing and purchasing broker contacts the payment processing function to charge the purchasing transaction to the customer and to credit the distributor.

One skilled in the art will recognize that other variations for processing ESD and non-ESD transactions would also operate with the licensing and purchasing broker. For example, instead of the user interface library offering related non-ESD merchandise, the WEB pages of the virtual store may offer both ESD and non-ESD items for purchase. In this scenario, a graphical icon (or similar object) associated with each non-ESD item available for purchase is displayed in addition to icons for ESD items. However, unlike the icons associated with ESD items, these icons are not linked to a download file that causes components to be downloaded, because online delivery is not possible. Instead, other virtual store code is linked to the non-ESD icons, which uses the purchasing library routines to send purchasing requests for non-ESD items to the licensing and purchasing broker.

Although specific embodiments of, and examples for, the present invention are described herein for illustrative purposes, it is not intended that the invention be limited to these embodiments. Equivalent methods, structures, processes, steps, and other modifications within the spirit of the invention fall within the scope of the invention. For example, the teachings provided herein of the present invention can be applied to other client/server architectures, not necessarily the exemplary Internet based, HTTP model described above. These and other changes may be made to the invention in light of the above detailed description. Accordingly, the invention is not limited by the disclosure, but instead the scope of the present invention is to be determined by the following claims.

What is claimed is:

1. A computer system for conducting electronic commerce, including:

a store computer that receives requests for electronic data from a client computer and that, in response to receiving the request, sends to the client computer a download component that coordinates the download of the electronic data;

a supplier computer that receives a request from the download component of the client computer to download the electronic data and that, in response to receiving the request, sends the electronic data and a licensing component to the client computer, the licensing component for coordinating the licensing of the electronic data; and

a licensing computer that receives a request from the licensing component of the client computer to license electronic data and that, in response to receiving the request, determines whether access to the electronic data is to be allowed at the client computer, and when access is allowed, sends a notification that access is allowed to the client computer.

2. The system of claim **1** wherein the licensing computer is for receiving a request from the licensing component for merchandise that is not transmitted online and for transmitting an order for physical shipment of the merchandise that is not transmitted online.

3. The system of claim **1** including a payment processing computer for processing payments for the electronic data.

4. The system of claim **1** wherein the store computer, the supplier computer, and the licensing computer are separate computers.

5. The system of claim **1** wherein the store computer, the supplier computer, and the licensing computer are separate web servers.

6. The system of claim **1** wherein the virtual store computer, the supplier computer, and the licensing computer are separate web sites.

7. A method in a computer system for conducting electronic commerce, including:

requesting a first web server to order electronic data;

receiving in response to the request a download component for coordinating the download of the electronic data; and

under control of the download component, downloading from a second web server the electronic data.

8. The method of claim **7** wherein the download component also downloads a licensing component and including:

under control of the licensing component, requesting and receiving from a third web server a license for using the electronic data; and

using the electronic data in accordance with the received license.

9. The method of claim **8** including:

under control of a payment component, authorizing payment for the electronic data.

10. The method of claim **7** wherein the downloaded electronic data is encrypted.

11. A method in a store computer for coordinating electronic commerce, the method including:

receiving from a client computer a request to purchase electronic data; and

in response to receiving the request, sending to the client computer a download component, the download component for coordinating the download of the electronic data from a supplier computer to the client computer, the supplier computer for downloading to the client computer the electronic data when requested by the download component.

12. The method of claim **11** wherein the supplier computer downloads a licensing component that requests a licensing computer for a license to use the electronic data.

13. The method of claim **11** wherein the store computer, the client computer, and the supplier computer communicate via the Internet.

14. A first computer for coordinating electronic commerce, including:

means for receiving from a second computer a request to purchase electronic data; and

means for, in response to receiving the request, sending to the second computer a download component, the download component for coordinating the download of the electronic data from a third computer to the second computer, the third computer for downloading to the second computer the electronic data when requested by the download component.

15. The first computer of claim **14** wherein the third computer downloads a licensing component that requests a fourth computer for a license to use the electronic data.

16. The first computer of claim **14** wherein the computers communicate via the Internet.

* * * * *

EXHIBIT H

US006141698A

United States Patent [19]

Krishnan et al.

[11] **Patent Number: 6,141,698**

[45] **Date of Patent: Oct. 31, 2000**

[54] **METHOD AND SYSTEM FOR INJECTING NEW CODE INTO EXISTING APPLICATION CODE**

[75] Inventors: **Ganapathy Krishnan**, Bellevue; **Scott Oyler**, Seattle, both of Wash.

[73] Assignee: **Network Commerce Inc.**, Seattle, Wash.

[21] Appl. No.: **08/792,719**

[22] Filed: **Jan. 29, 1997**

[51] **Int. Cl.**[7] **G06F 9/46**

[52] **U.S. Cl.** **709/331**

[58] **Field of Search** 709/300–305, 709/331; 713/200, 201, 202; 380/4; 717/4, 11

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,103,476	4/1992	Waite et al.	380/4
5,193,180	3/1993	Hastings	395/575
5,335,344	8/1994	Hastings	395/575
5,375,241	12/1994	Walsh	395/700
5,381,547	1/1995	Flug et al.	395/700
5,535,329	7/1996	Hastings	365/183.11
5,539,908	7/1996	Chen et al.	395/700
5,548,759	8/1996	Lipe	395/600
5,563,946	10/1996	Cooper et al.	380/4
5,577,120	11/1996	Penzias	380/23
5,594,903	1/1997	Bunnell et al.	395/712
5,604,803	2/1997	Aziz	380/25
5,649,099	7/1997	Theimer et al.	395/187.01
5,659,614	8/1997	Bailey, III	380/4
5,675,645	10/1997	Schwartz et al.	380/4
5,689,560	11/1997	Cooper et al.	380/4
5,892,904	4/1999	Atkinson et al.	713/201
5,925,117	7/1999	Kirby et al.	701/101
5,953,534	9/1999	Romer et al.	717/11
5,974,549	10/1999	Golan	713/200
5,999,622	12/1999	Yasukawa et al.	380/4
6,027,235	2/2000	Shaughnessy	717/4

FOREIGN PATENT DOCUMENTS

0 367 700 A2	5/1990	European Pat. Off.	G06F 1/00
0 667 572 A1	8/1995	European Pat. Off.	G06F 9/445

OTHER PUBLICATIONS

Pietrek, Matt Learn System–Level Win32 Coding Techniques by Writing an API Spy Program, Microsoft Systems Journal, p. (22), Dec. 1994.

Petzold, Charles, *Programming Windows*, 2d ed., Microsoft Press, Redmond, 1990, pp. 877–915.

Microsoft, *Microsoft Portable Executable and Common Object File Format*, Specification 4.1, Microsoft Corporation, Aug. 1994.

Matt Pietrek, "Peering Inside the PE: A Tour of the *Win32* Portable Executable File Format," *Microsoft Systems Journal*, Mar. 1994.

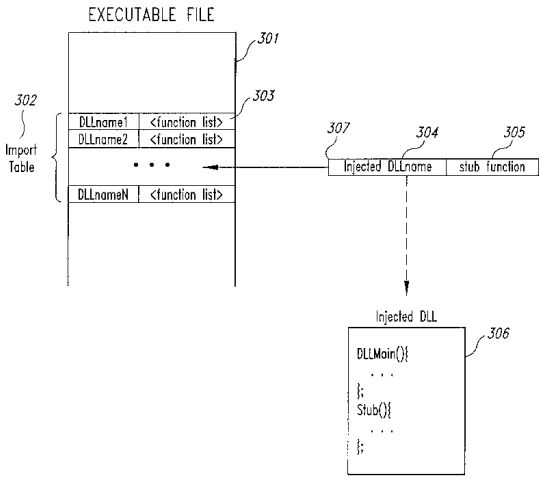
Schneier, Bruce, *Applied Cryptography*, 2d ed., John Wiley & Sons, 1996.

Primary Examiner—Alvin E. Oberley
Assistant Examiner—St. John Courtenay, III
Attorney, Agent, or Firm—Perkins Coie LLP

[57] **ABSTRACT**

A method and system for modifying the behavior of existing executable code by injecting new code into an executable file is provided. The injection mechanism injects a reference to new code contained in a DLL into an existing executable file such that, when the code of the executable file is executed, the DLL is automatically loaded and the new code is automatically executed. A reference to the DLL is injected into the executable file by either modifying an import table of the file, which causes automatic loading of the DLLs referred to therein, or by adding DLL loader code to the file. The DLLs loader code uses an underlying operating system call to load the DLL. Further, the injection mechanism provides enhanced security by injecting security code and data into the executable file. The injected security code mechanism uses an incremental encryption and decryption process to encrypt and decrypt portions of the executable file in a more secure manner.

26 Claims, 14 Drawing Sheets



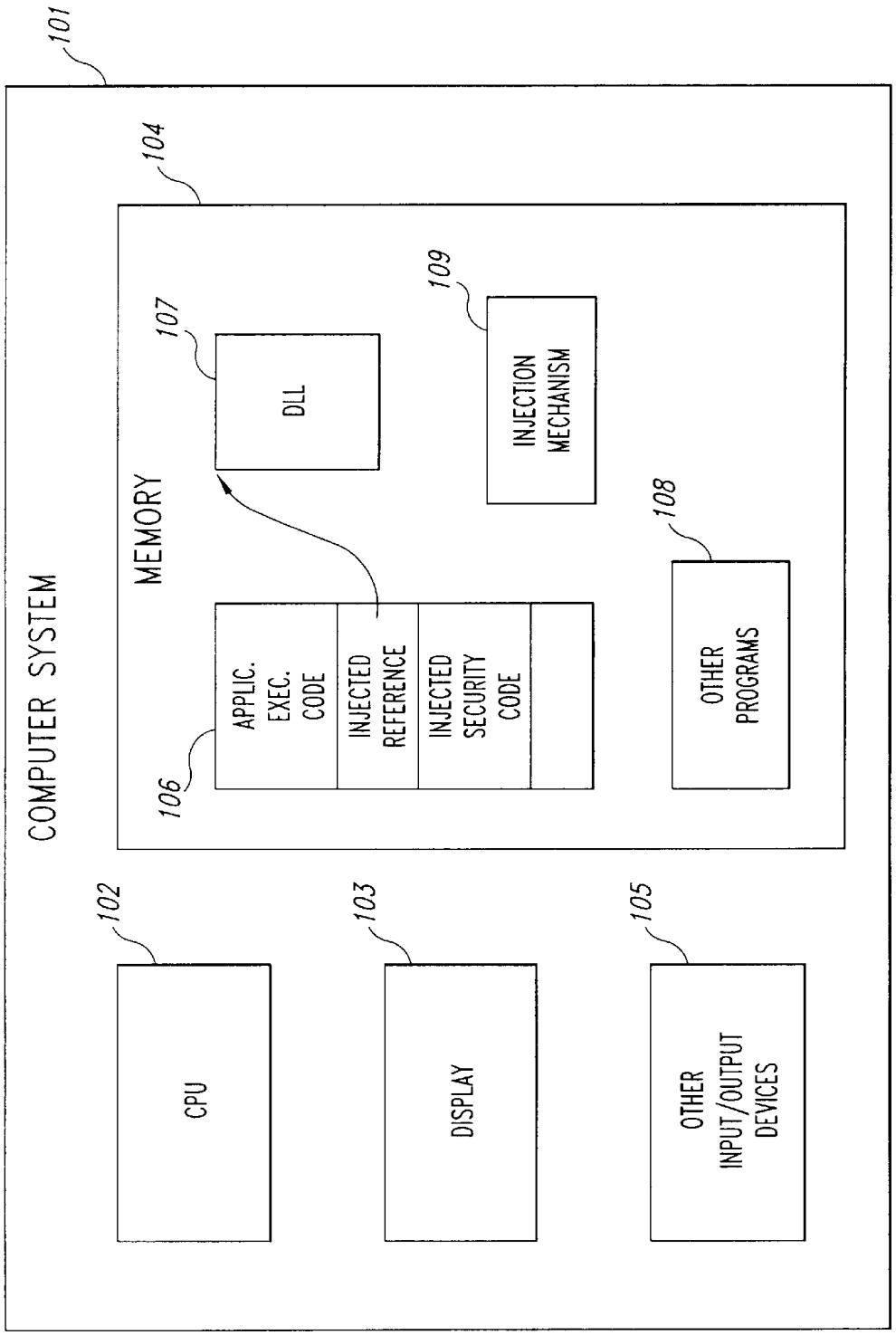

101
COMPUTER SYSTEM
104
MEMORY
107
DLL
109
INJECTION MECHANISM
108
OTHER PROGRAMS
106
APPLIC. EXEC. CODE
INJECTED REFERENCE
INJECTED SECURITY CODE
102
CPU
103
DISPLAY
105
OTHER INPUT/OUTPUT DEVICES

Fig. 1

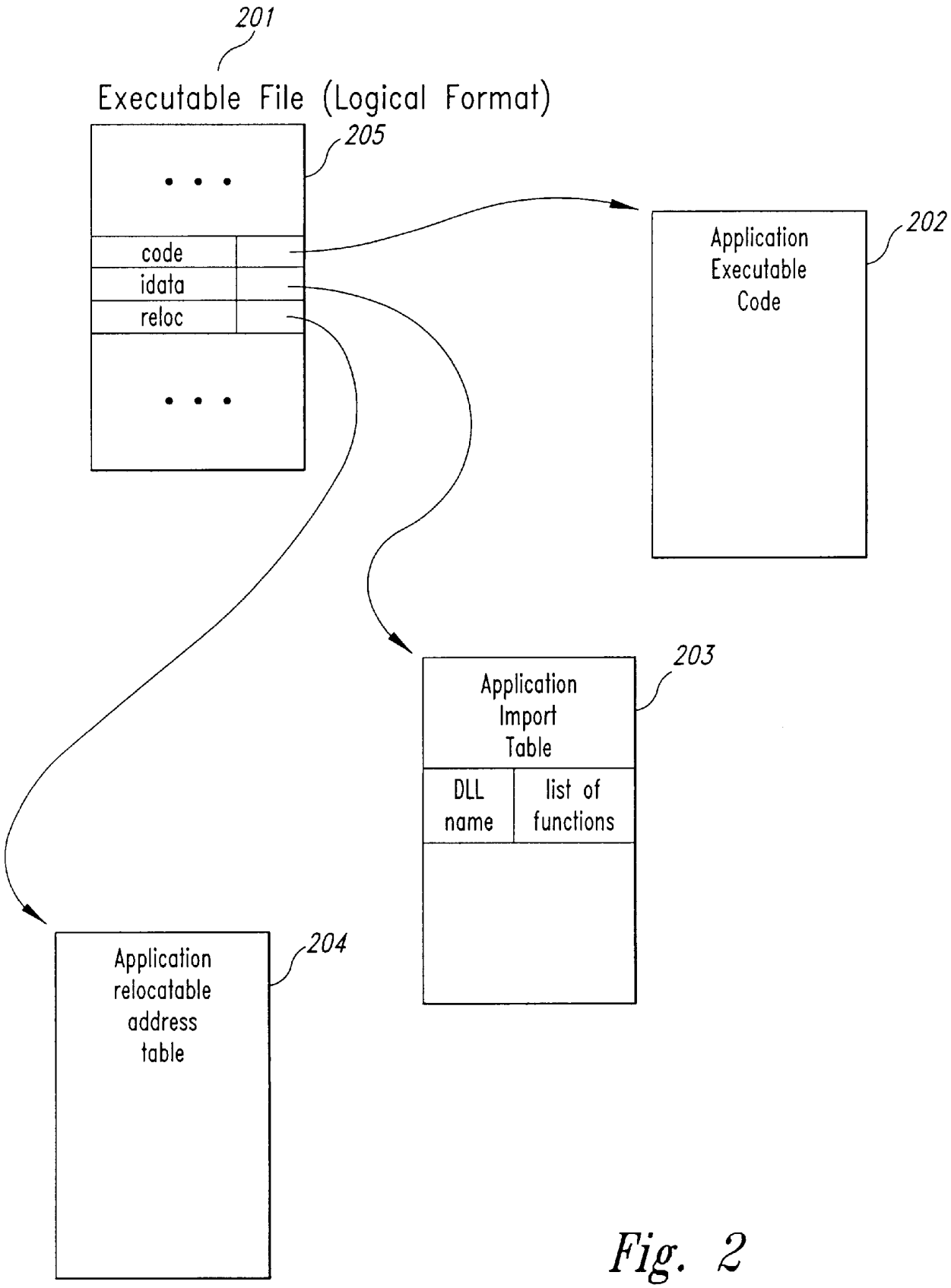

201
Executable File (Logical Format)
205
code
idata
reloc
202
Application Executable Code
203
Application Import Table
DLL name
list of functions
204
Application relocatable address table

Fig. 2

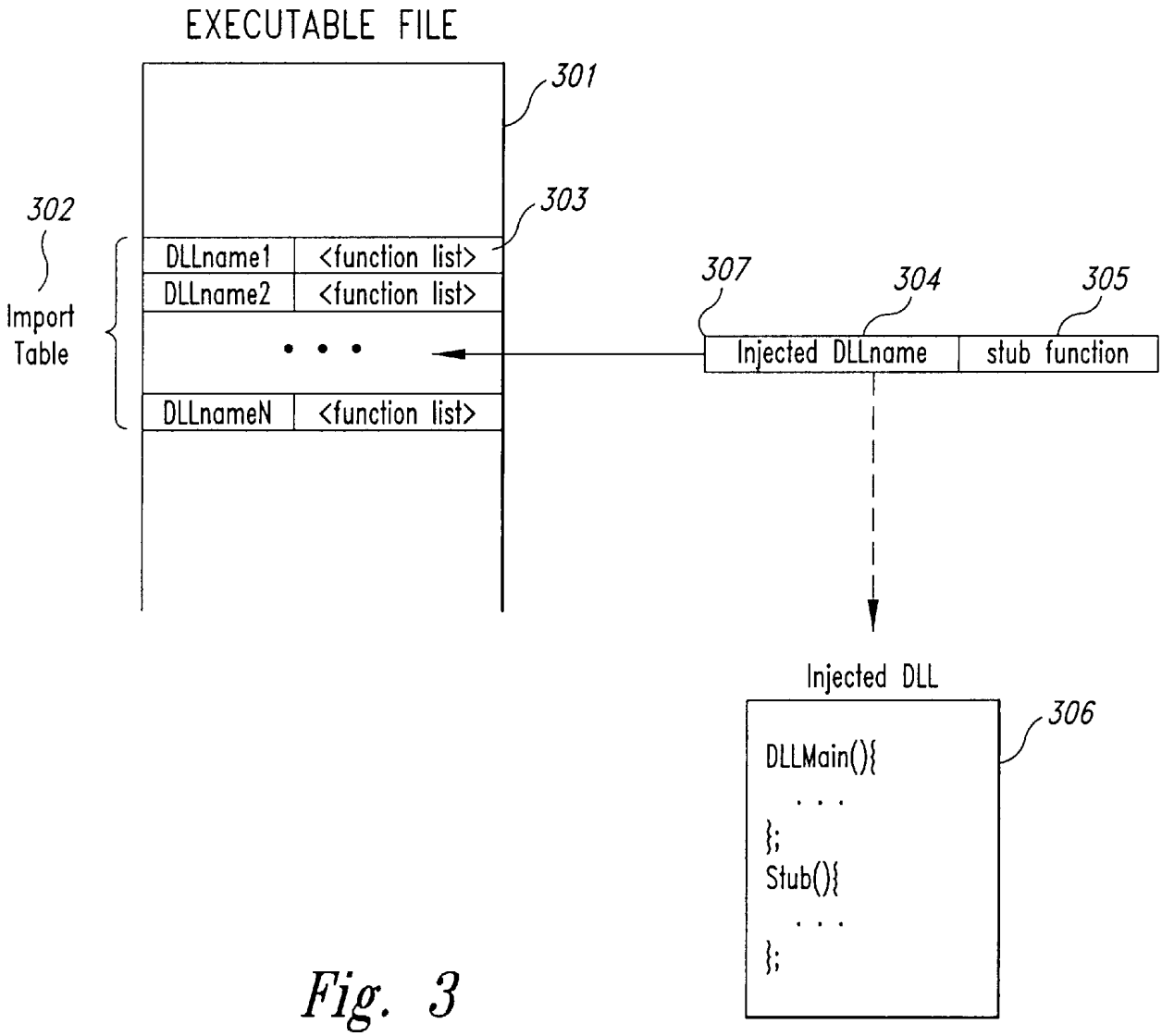

EXECUTABLE FILE
301
302
303
Import Table
DLLname1
<function list>
DLLname2
<function list>
DLLnameN
<function list>
307
304
305
Injected DLLname
stub function
Injected DLL
306
DLLMain(){
};
Stub(){
};

Fig. 3

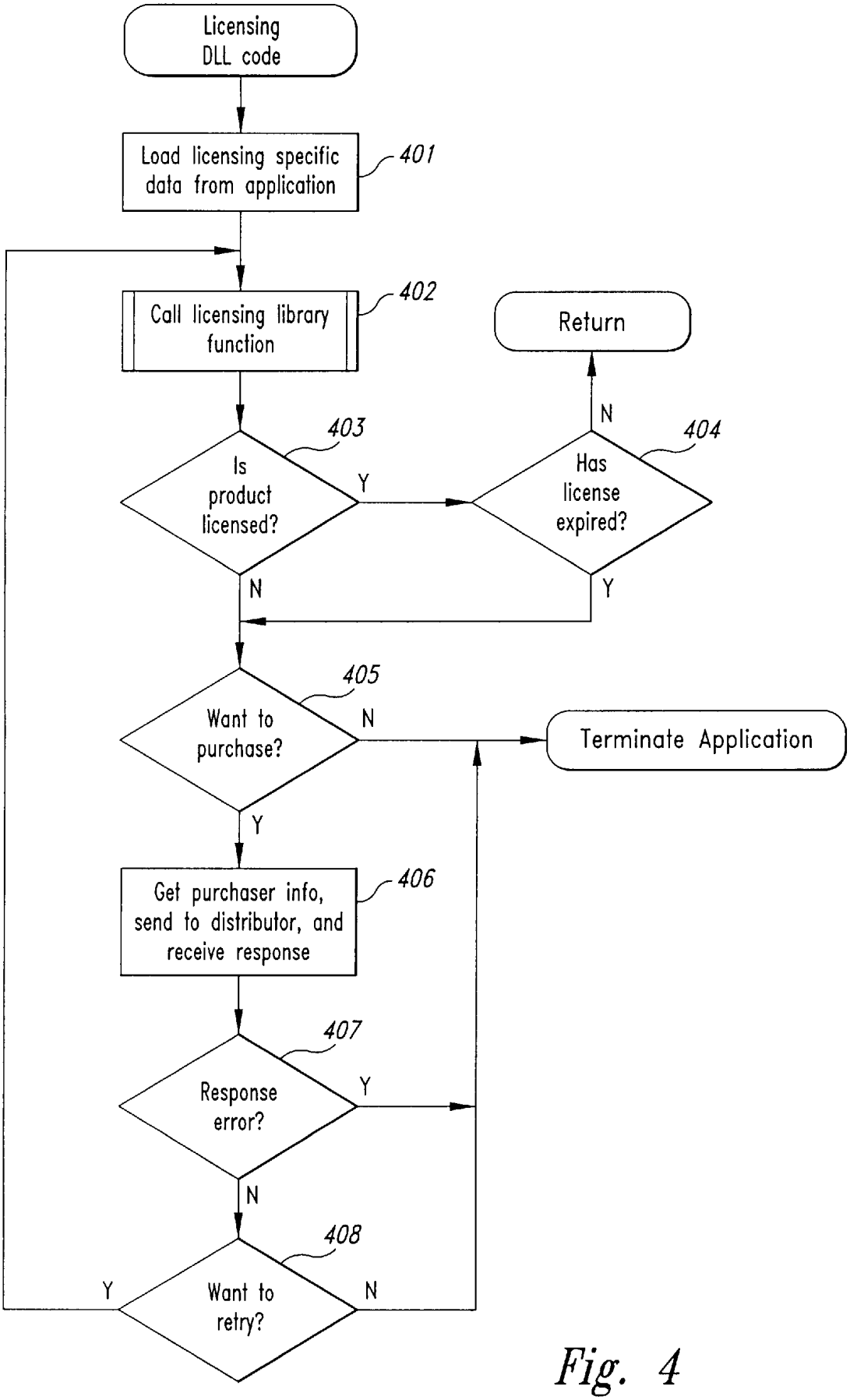

Licensing DLL code
Load licensing specific data from application
401
Call licensing library function
402
Return
403
Is product licensed?
Y
404
Has license expired?
N
Y
N
405
Want to purchase?
N
Terminate Application
Y
Get purchaser info, send to distributor, and receive response
406
407
Response error?
Y
N
408
Want to retry?
Y
N

Fig. 4

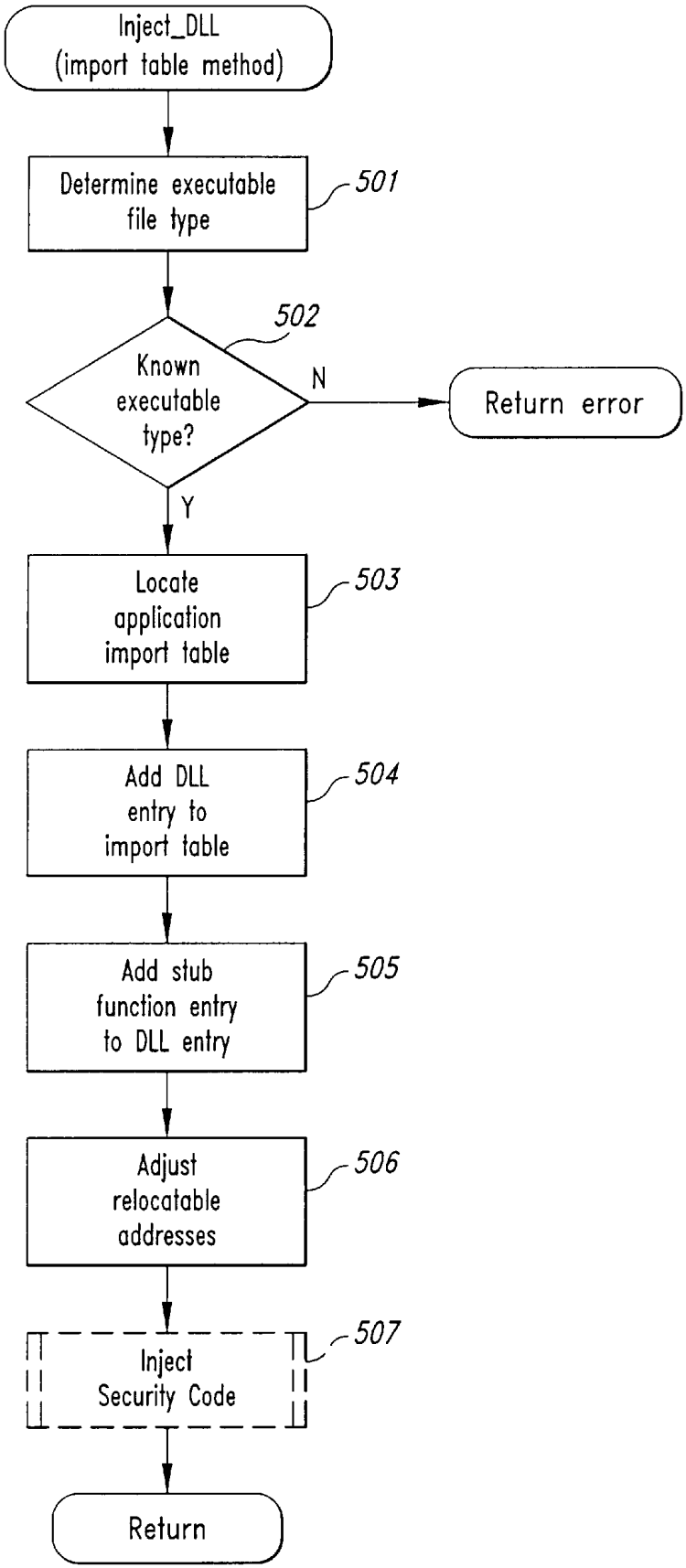

Inject_DLL
(import table method)
Determine executable file type
501
Known executable type?
502
N
Return error
Y
Locate application import table
503
Add DLL entry to import table
504
Add stub function entry to DLL entry
505
Adjust relocatable addresses
506
Inject Security Code
507
Return

Fig. 5

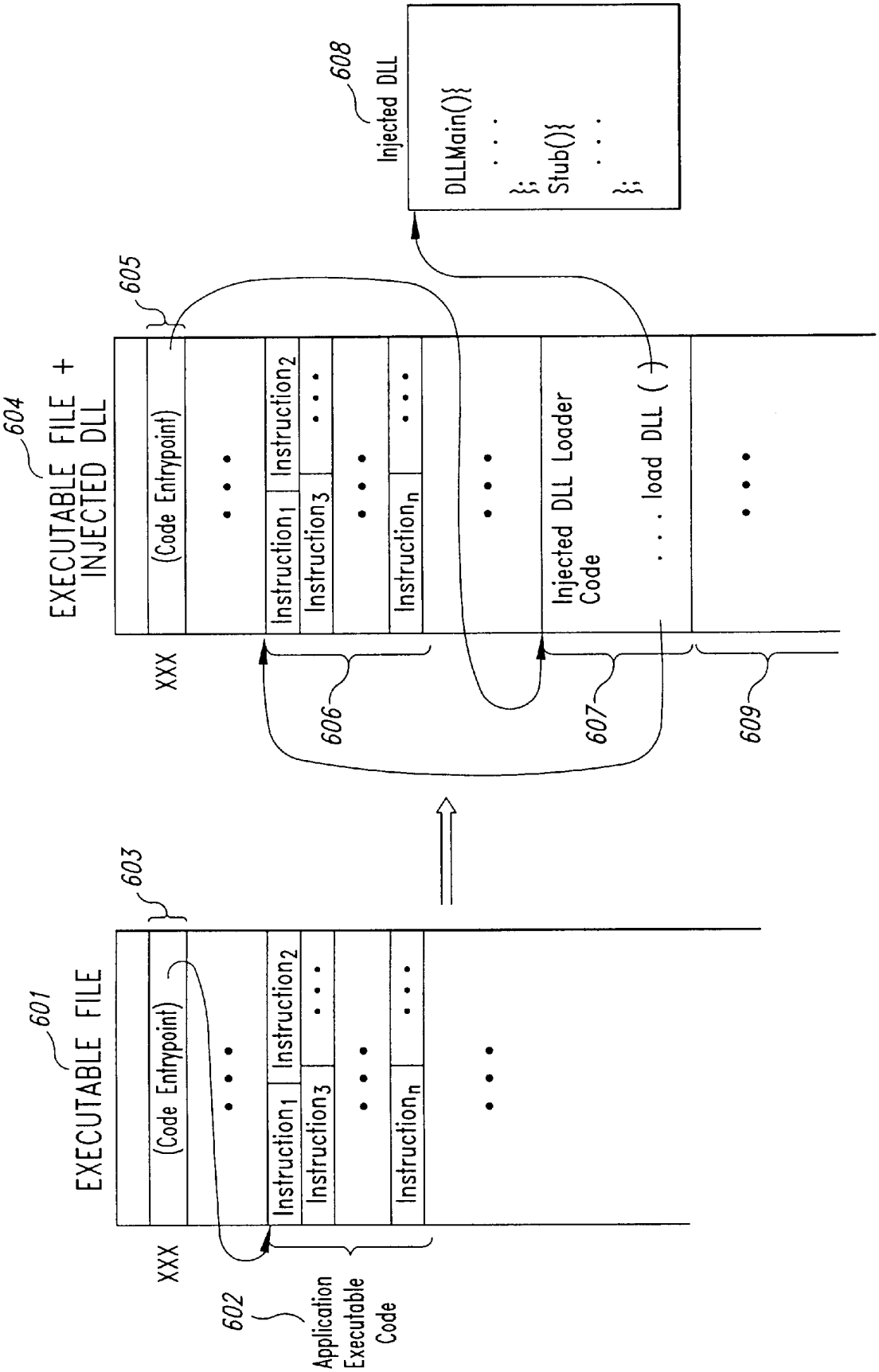

608
Injected DLL
DLLMain(){}
Stub(){}
605
604
EXECUTABLE FILE + INJECTED DLL
(Code Entrypoint)
Instruction1
Instruction2
Instruction3
Instructionn
Injected DLL Loader Code
load DLL ( )
XXX
606
607
609
601
EXECUTABLE FILE
603
602
Application Executable Code

Fig. 6

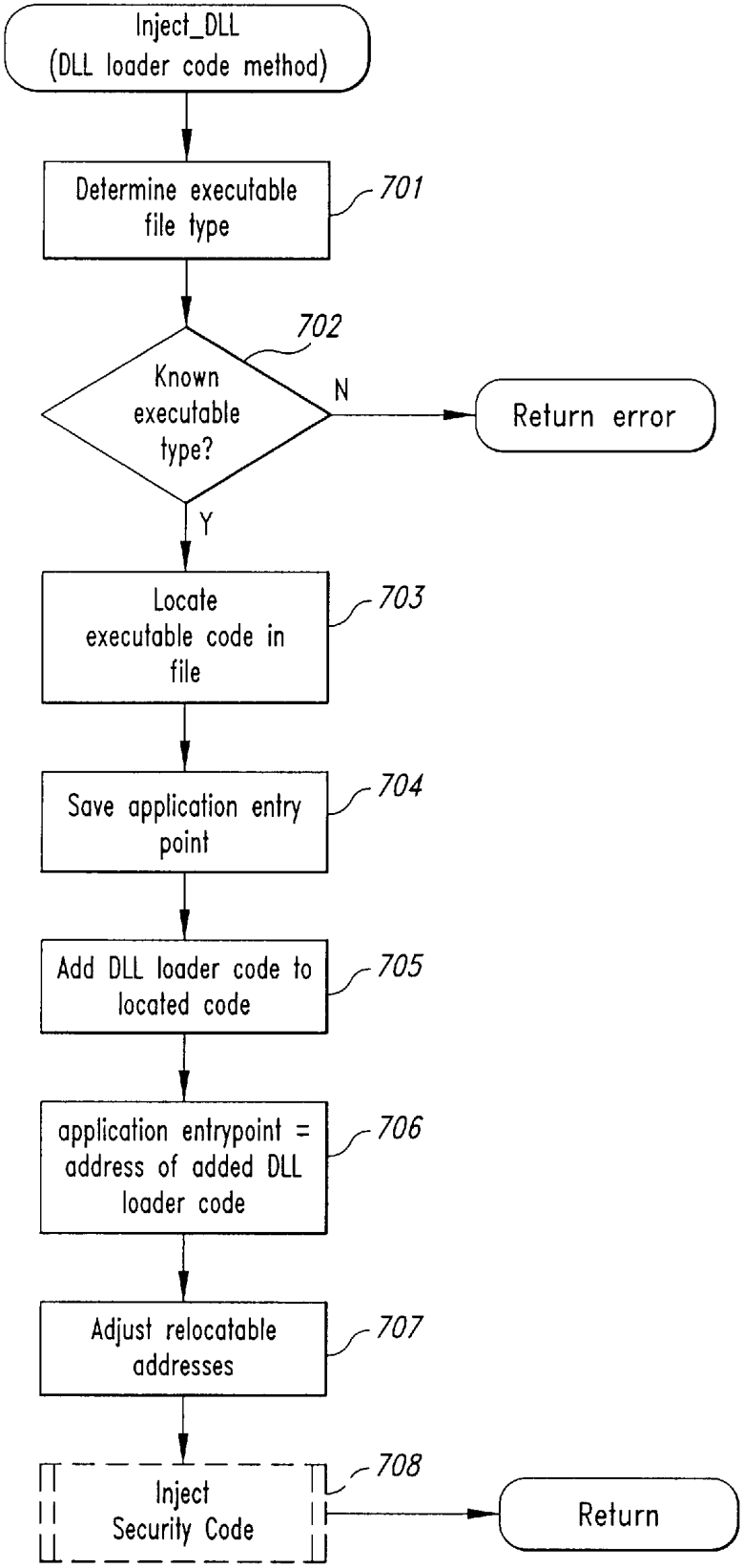

Inject_DLL
(DLL loader code method)
Determine executable file type
701
Known executable type?
702
N
Return error
Y
Locate executable code in file
703
Save application entry point
704
Add DLL loader code to located code
705
application entrypoint = address of added DLL loader code
706
Adjust relocatable addresses
707
Inject Security Code
708
Return

Fig. 7

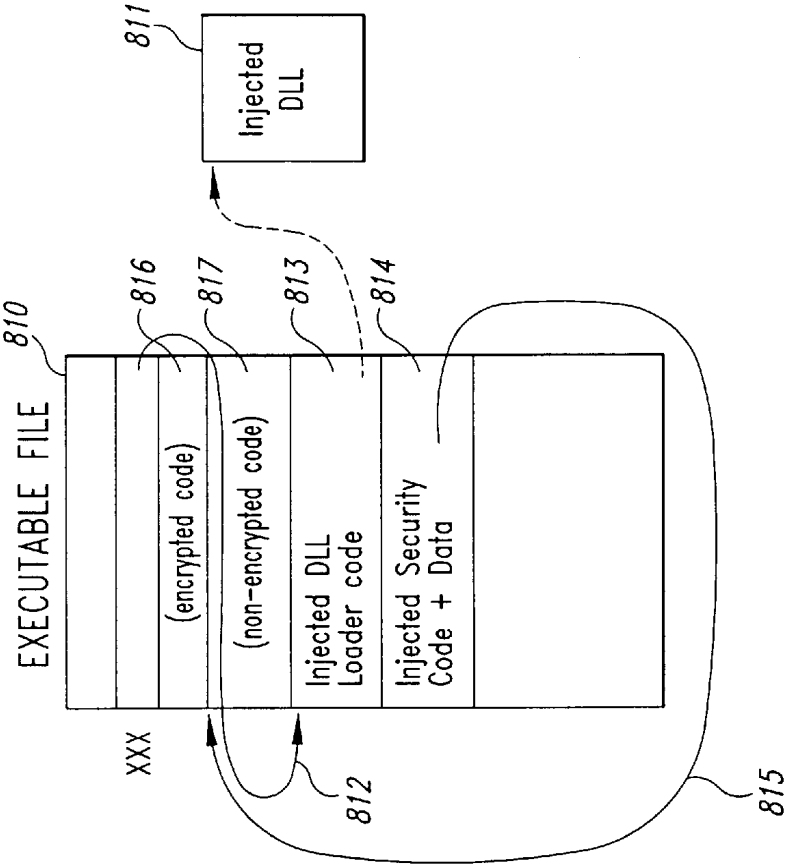

811
Injected DLL
816
817
813
814
810
EXECUTABLE FILE
(encrypted code)
(non-encrypted code)
Injected DLL Loader code
Injected Security Code + Data
XXX
812
815

Fig. 8B

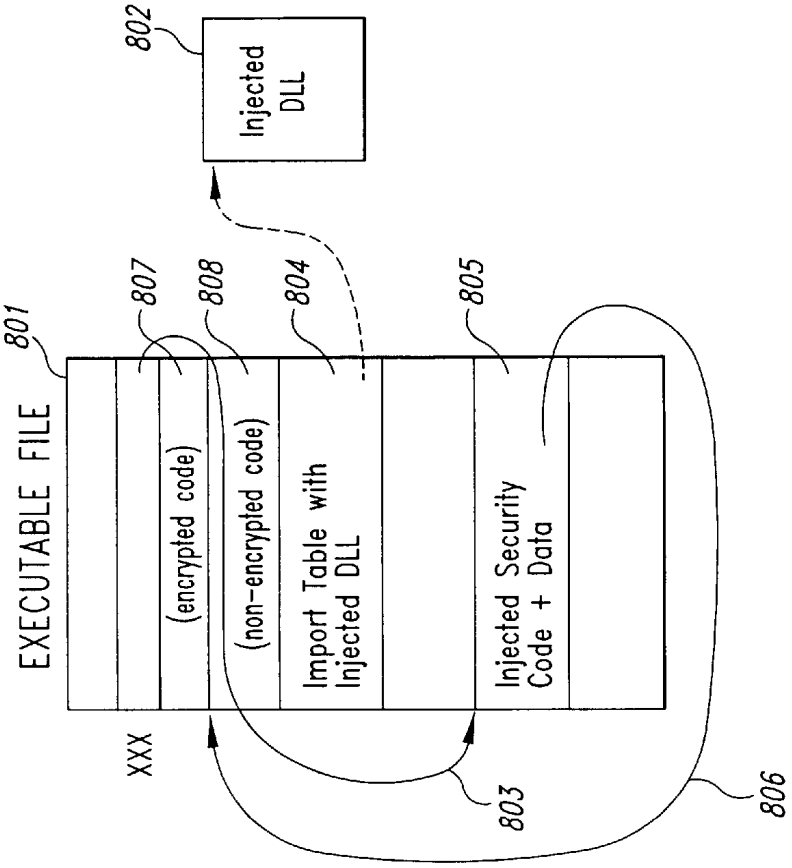

802
Injected DLL
807
808
804
805
801
EXECUTABLE FILE
(encrypted code)
(non-encrypted code)
Import Table with Injected DLL
Injected Security Code + Data
XXX
803
806

Fig. 8A

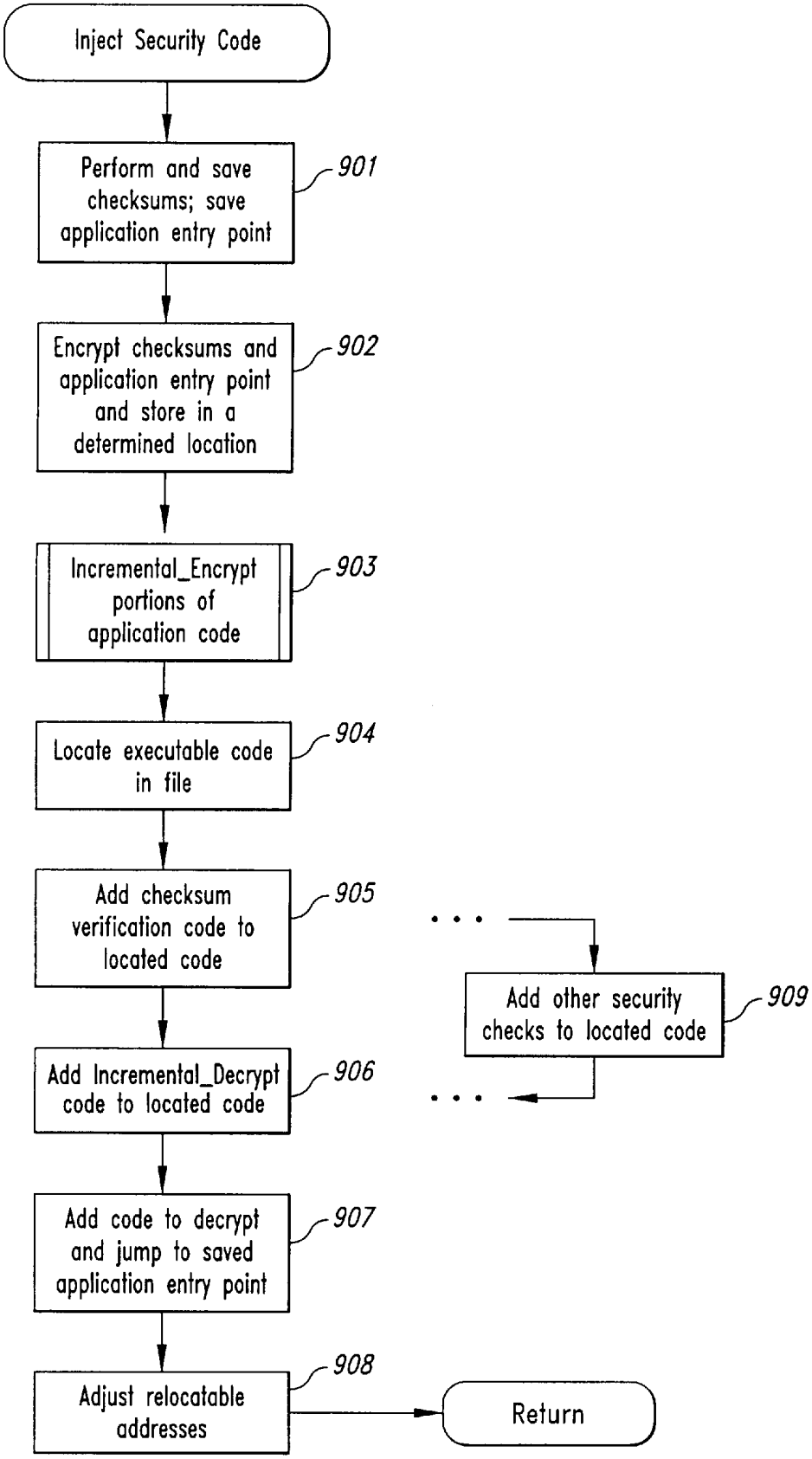

Inject Security Code
Perform and save checksums; save application entry point
901
Encrypt checksums and application entry point and store in a determined location
902
Incremental_Encrypt portions of application code
903
Locate executable code in file
904
Add checksum verification code to located code
905
Add Incremental_Decrypt code to located code
906
Add code to decrypt and jump to saved application entry point
907
Adjust relocatable addresses
908
Return
Add other security checks to located code
909

Fig. 9

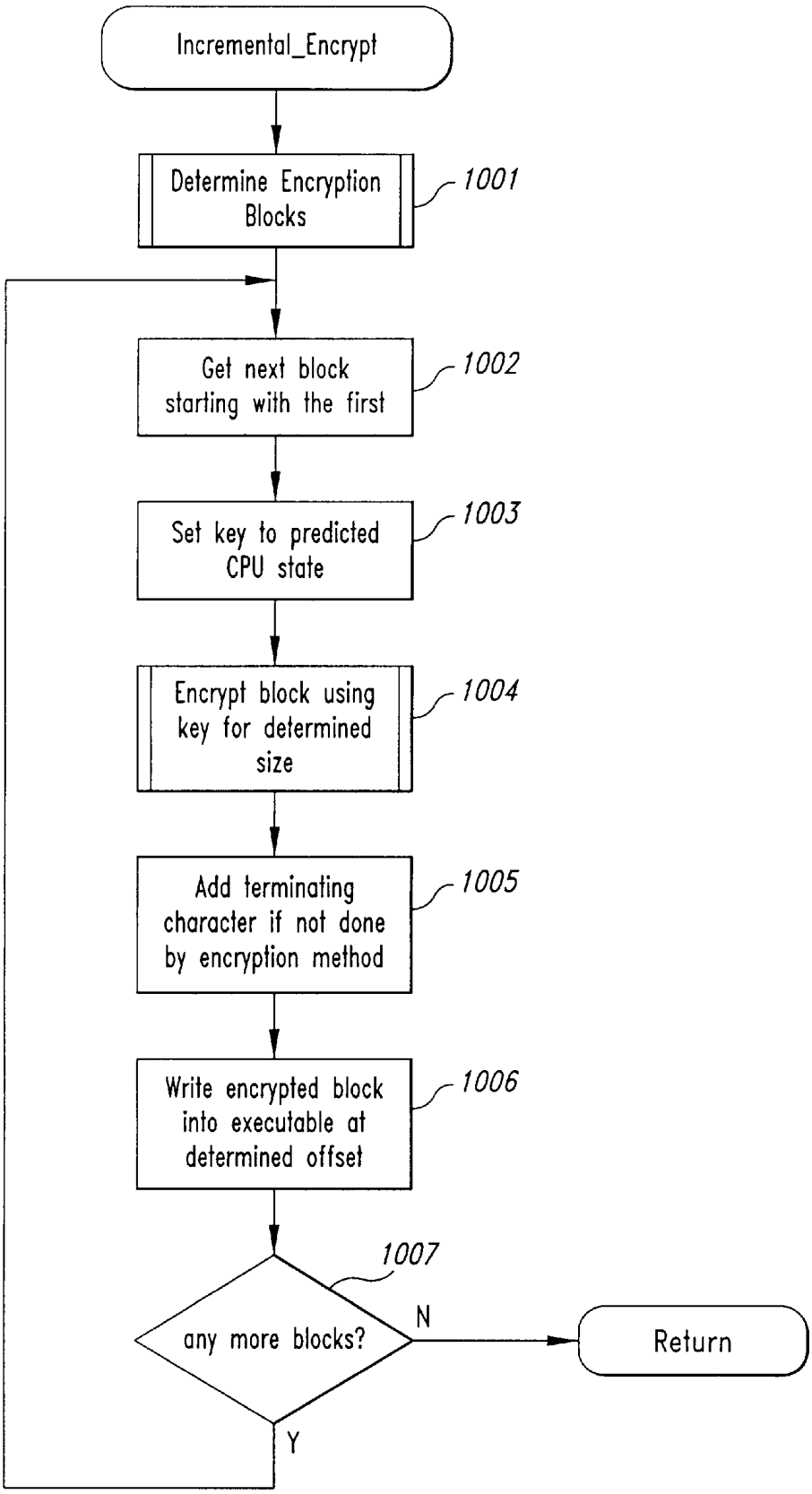

Incremental_Encrypt
Determine Encryption Blocks
1001
Get next block starting with the first
1002
Set key to predicted CPU state
1003
Encrypt block using key for determined size
1004
Add terminating character if not done by encryption method
1005
Write encrypted block into executable at determined offset
1006
1007
any more blocks?
N
Return
Y

Fig. 10

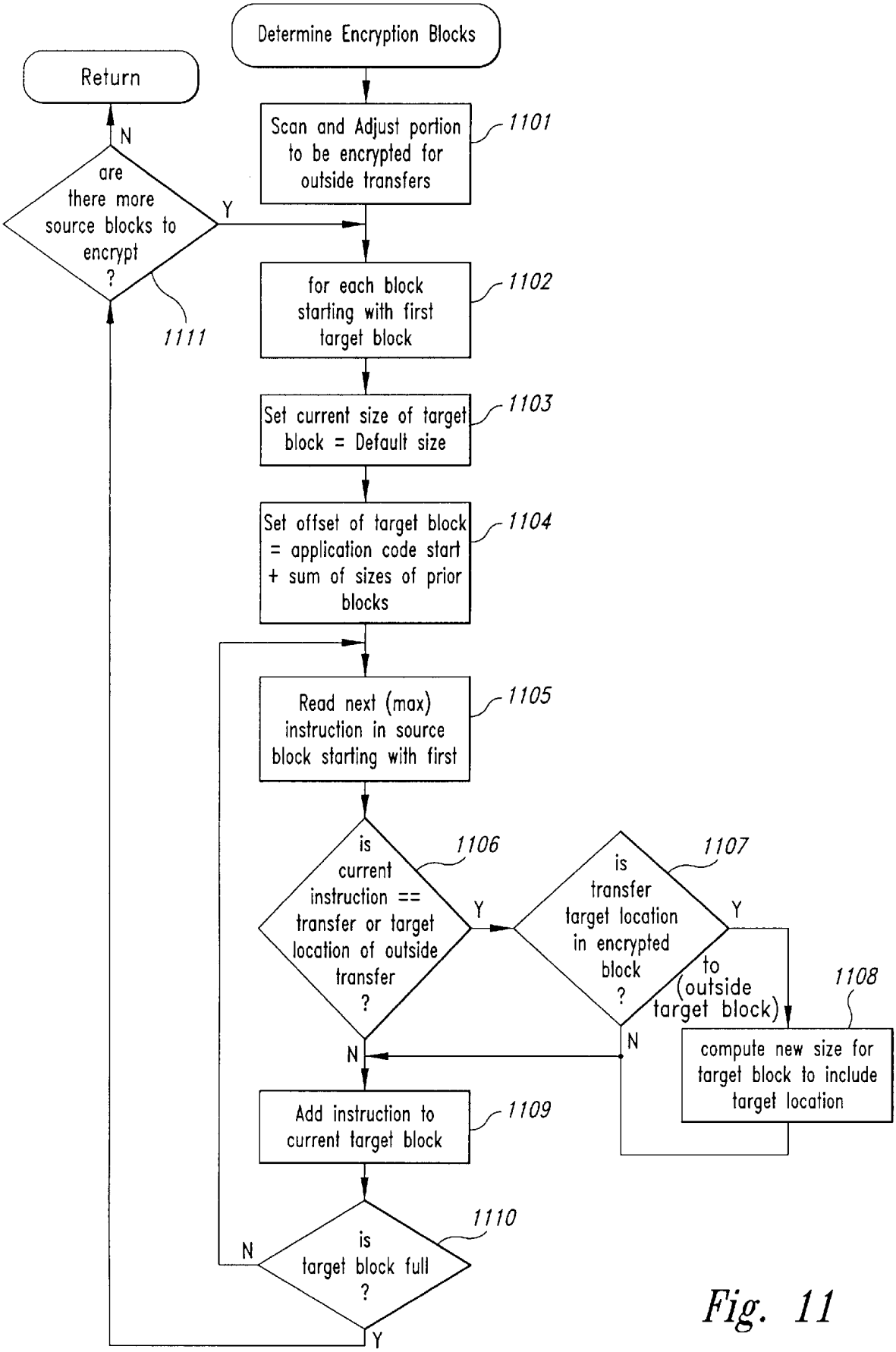

Determine Encryption Blocks
Return
Scan and Adjust portion to be encrypted for outside transfers
1101
are there more source blocks to encrypt ?
N
Y
1111
for each block starting with first target block
1102
Set current size of target block = Default size
1103
Set offset of target block = application code start + sum of sizes of prior blocks
1104
Read next (max) instruction in source block starting with first
1105
is current instruction == transfer or target location of outside transfer ?
1106
Y
N
is transfer target location in encrypted block ?
1107
Y
N
to (outside target block)
1108
compute new size for target block to include target location
Add instruction to current target block
1109
is target block full ?
1110
N
Y

Fig. 11

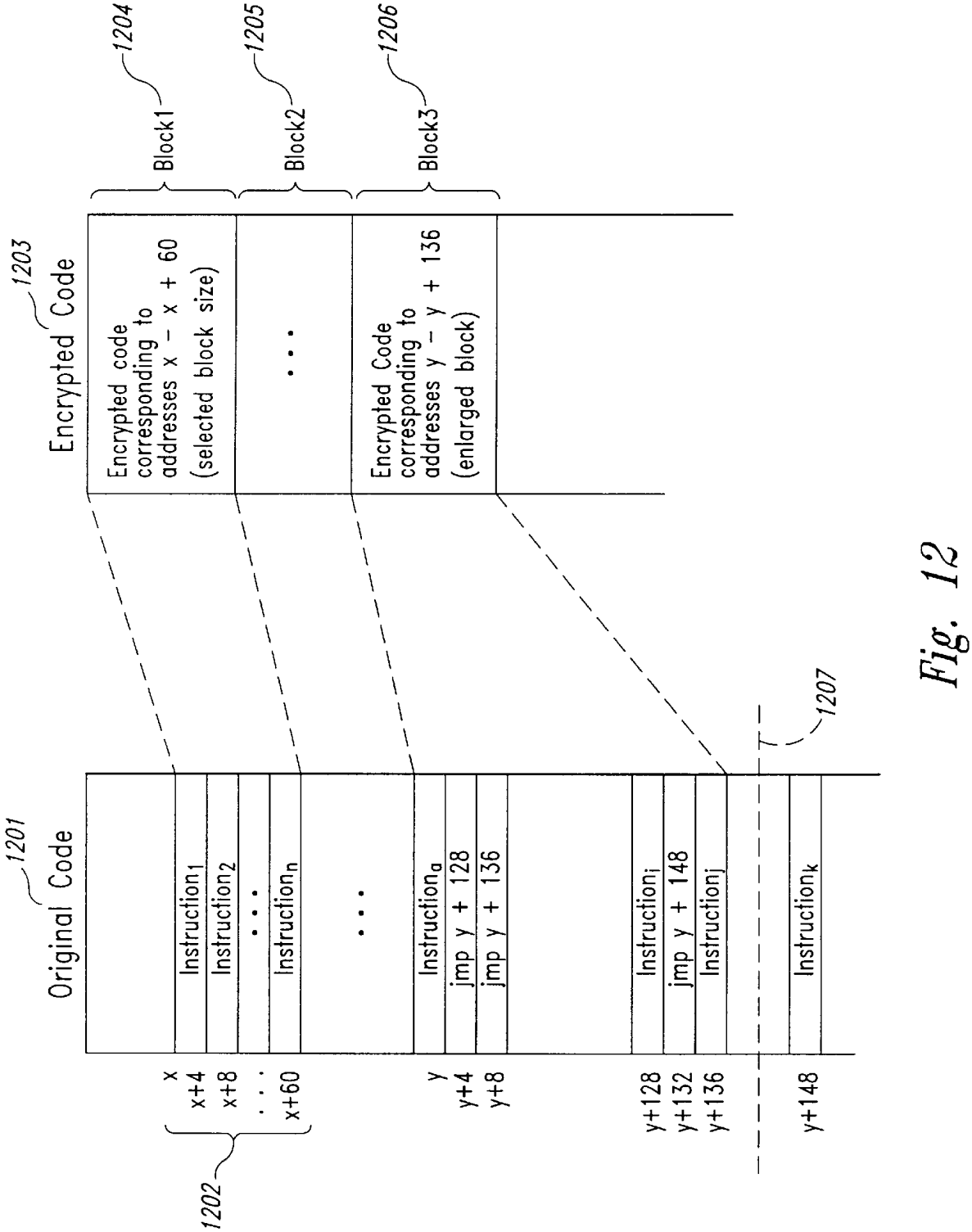

1204
1205
1206
Block1
Block2
Block3
1203
Encrypted Code
Encrypted code corresponding to addresses x – x + 60 (selected block size)
Encrypted Code corresponding to addresses y – y + 136 (enlarged block)
1201
Original Code
Instruction1
Instruction2
Instructionn
Instructiona
jmp y + 128
jmp y + 136
Instructioni
jmp y + 148
Instructionj
Instructionk
1207
x
x+4
x+8
x+60
y
y+4
y+8
y+128
y+132
y+136
y+148
1202

Fig. 12

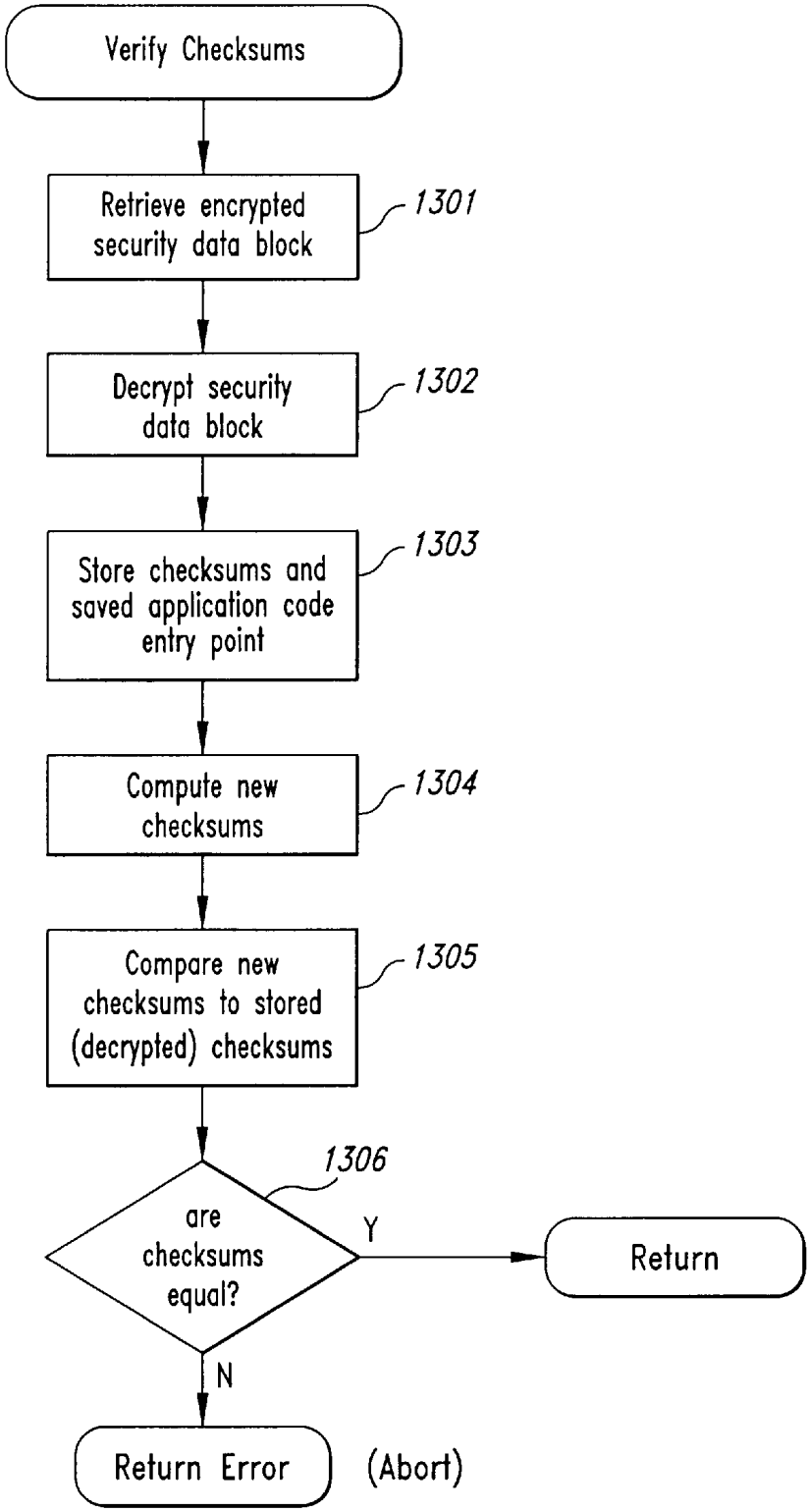

Verify Checksums
Retrieve encrypted security data block
1301
Decrypt security data block
1302
Store checksums and saved application code entry point
1303
Compute new checksums
1304
Compare new checksums to stored (decrypted) checksums
1305
1306
are checksums equal?
Y
Return
N
Return Error
(Abort)

Fig. 13

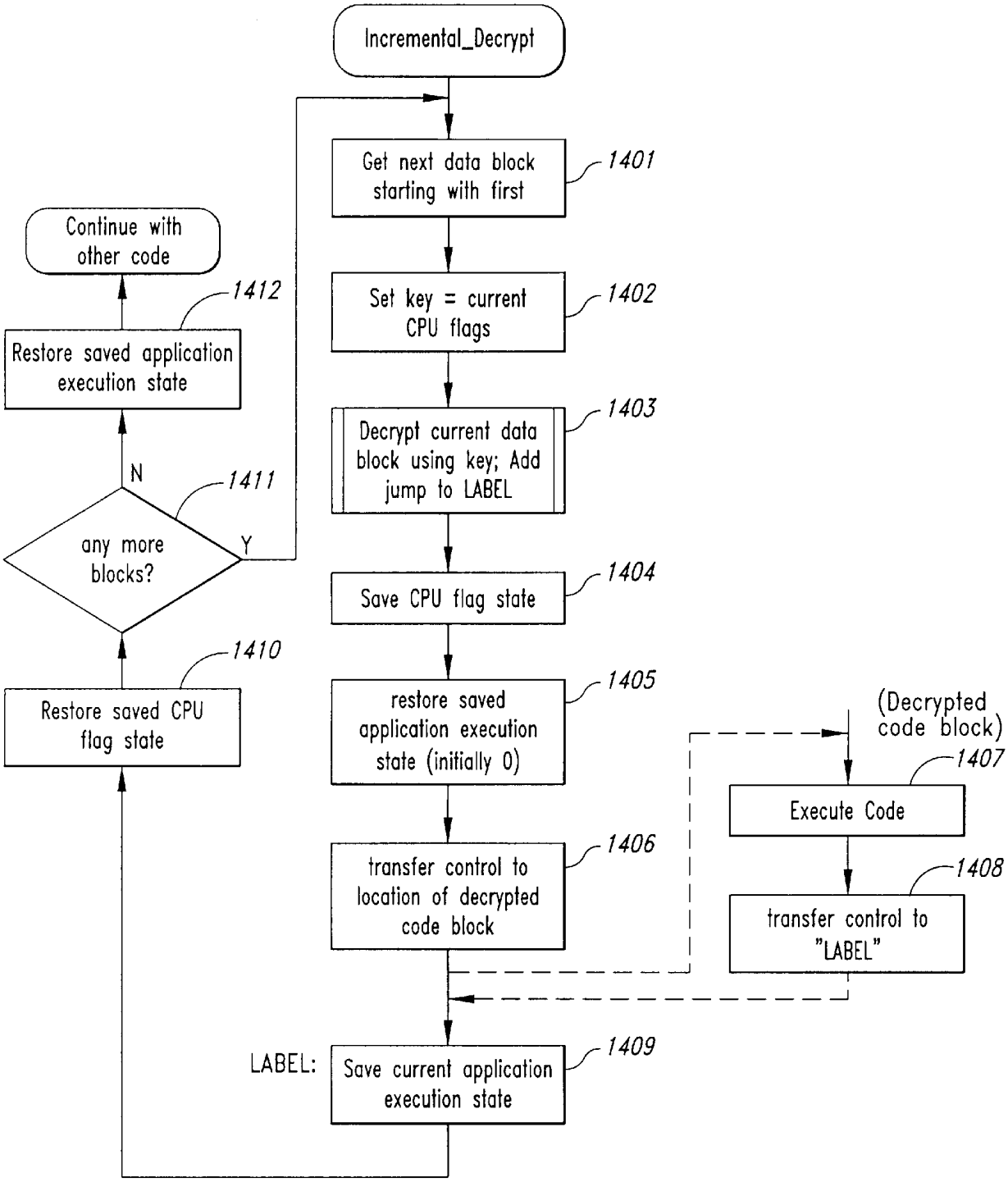

Incremental_Decrypt
Get next data block starting with first
1401
Set key = current CPU flags
1402
Decrypt current data block using key; Add jump to LABEL
1403
Save CPU flag state
1404
restore saved application execution state (initially 0)
1405
transfer control to location of decrypted code block
1406
LABEL:
Save current application execution state
1409
any more blocks?
1411
N
Y
Restore saved CPU flag state
1410
Restore saved application execution state
1412
Continue with other code
(Decrypted code block)
Execute Code
1407
transfer control to "LABEL"
1408

Fig. 14

METHOD AND SYSTEM FOR INJECTING NEW CODE INTO EXISTING APPLICATION CODE

TECHNICAL FIELD

The present invention relates to modifying existing application code and, in particular, to injecting a dynamic link library into an existing executable file.

BACKGROUND OF THE INVENTION

In current computer systems, there often exists a need for modifying the behavior of executable code stored in a pre-existing executable file. For the purposes of this application, an "executable file" is any type of code image and is not limited to a particular type of executable file or a file with a particular file name extension. In particular, the need exists to change the behavior of an application without recompiling the application. This need is especially apparent in situations where it is impossible or too much work to recompile the application. For example, an application may be developed by a source company at one site and distributed to a third party vendor at another site. The third party vendor may wish to incorporate vendor-specific code into the application before redistributing it to an end customer. However, the third party vendor may not have access to the source code that the source company used to generate the executable file. Thus, the third party vendor cannot change and recompile the source code to generate a new executable file with the vendor-specific code.

As another example, especially relevant in today's extensive networking environments, a company may desire to put an existing application on the Internet and somehow incorporate licensing code to limit any use of illegal copies of the application. Current systems have tried various solutions to incorporate licensing code into an existing application. According to one technique, which will be referred to herein as "wrapping," a second application program (a wrapper program) is distributed on the network, which includes an encrypted version of the original application program. The wrapper program, when installed, decrypts the encrypted original application program and then proceeds to execute the original application program. To successfully decrypt the program, a legitimate end user must provide the proper licensing information to enable the decryption to operate. A security hole exists, however, in that, while the wrapping program is in the process of decrypting the original application executable file, temporary files are created to hold the decrypted program code. Once the entire original application program has been decrypted and stored in the temporary file, a "software pirate" can then make multiple copies of the original unencrypted application program in the temporary file and can distribute them illegally.

Further, use of the wrapping technique to incorporate licensing provides only limited additional security to a vendor who implements what is known as a "try and buy" licensing program. A try and buy licensing program typically distributes an application program with either limited functionality or for a limited time of use to enable a potential customer to explore the application. Functionality is typically limited, for example, by turning off a set of features. Once the potential customer is satisfied, the customer can pay for and license the application program properly. If an application program is distributed using the wrapping technique to potential customers for the purpose of a try and buy program, then, when the program is decrypted and stored in a temporary file, a software pirate can determine how to turn on the disabled features or how to remove the license expiration data. These security problems can result in the distribution of illegal copies, which are hard to detect and monitor in a global network environment.

A second technique for modifying the behavior of an existing application program directly inserts the new executable code into the executable file. Using the direct insertion method, an application developer determines where in the executable file the new code should be placed and inserts the code into the executable. After inserting the new code into the existing executable file, the application developer adjusts addresses that reference any relocatable code or data that follows the inserted code to account for the newly added code. However, it is very difficult for an application developer to determine where to insert code and to then test the entire application to ensure it works correctly. An application developer would typically need to disassemble the executable file and study the disassembled code to determine where to insert the code. Such disassembling and studying is a very time-consuming process. Furthermore, the process must be repeated for each application program, and for each version of each application program in which the code is to be inserted.

Thus, the need exists to modify the behavior of executable code stored in an existing executable file in a manner that is secure and that requires minimal testing outside the scope of standalone testing of the code that provides the modified behavior.

SUMMARY OF THE INVENTION

The present invention provides a method and system for injecting new code into already existing executable code within an executable file. The injection mechanism provided by the present invention can be used to inject a dynamic link library (DLL) that contains the new code or to inject arbitrary code into an existing executable file. The injection of new code enables the existing executable code to perform new behaviors. For example, licensing procedures can be added to an existing application by injecting a licensing DLL into the application using the injection mechanism.

In one embodiment, the injection mechanism injects into the existing executable file new DLL code and optionally injects additional security code, which is provided by the injection mechanism. Preferably, the injected security code performs checksum comparisons on some portions of the executable file, decrypts and executes a previously encrypted portion of the executable code, and decrypts and transfers execution control to a previously encrypted location in the original executable code. The injection of security code helps to prevent modification of the executable file to omit the injected code and thereby restore the executable file to its original, unmodified state. In the case of newly added licensing code, the injected security code aids in preventing illegal altering, copying, and distribution of the original executable code.

The injection mechanism provides two methods for injecting a DLL into existing executable code. The first method modifies an import table of the executable file to include a reference to the new DLL code. A second method modifies the executable file to include DLL, loader code, which is provided by the injection mechanism. The DLL loader code uses system provided calls to load the desired new DLL. The injection of security code can be utilized with both methods of injecting a DLL.

The present invention also provides incremental encryption and decryption techniques, which can be used to further

secure any of the injected code. The incremental encryption and decryption techniques operate by encrypting (and subsequently decrypting) blocks of code of varying sizes, using a different key for each block. The decryption code decrypts each block and executes the decrypted code, one block at a time, and overwrites each decrypted block when decrypting a next block. This process ensures that the entire nencrypted code is never visible at any one time.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a general purpose computer system for practicing embodiments of the injection mechanism.

FIG. **2** is a block diagram of a logical format of an executable file that can be used with the present invention.

FIG. **3** is an overview block diagram of the procedure for

FIG. **4** is a flow diagram of example code that can be placed into an injectable DLL in order to incorporate licensing into an existing application.

FIG. **5** is a detailed flow diagram of the steps used by the injection mechanism to inject a new DLL using the import table technique.

FIG. **6** is an overview block diagram of the modifications made to an executable file by the injection mechanism to inject a reference to a new DLL using the DLL loader code technique.

FIG. **7** is a flow diagram of the steps used by the injection mechanism to inject a new DLL using the DLL loader code technique.

FIG. **8**A is a block diagram of the logical layout of an executable file after security code has been injected into the executable file using the import table technique.

FIG. **8**B is a block diagram of the logical layout of an executable file after security code has been injected into the executable file using the DLL loader code technique.

FIG. **9** is an overview flow diagram of the steps performed by the injection mechanism for injecting security code and data into an executable file.

FIG. **10** is a flow diagram of the steps executed by an incremental encryption routine.

FIG. **11** is a detailed flow diagram of the steps performed by the injection mechanism to determine the number and size of the encryption blocks of data to be encrypted using the incremental encryption technique.

FIG. **12** is an example block diagram of the results of determining the size of encryption blocks according to the technique of FIG. **11**.

FIG. **13** is a detailed flow diagram of the verify checksum code added by the injection mechanism to an executable file.

FIG. **14** is a detailed flow diagram of the steps performed by incremental decryption.

DETAILED DESCRIPTION OF THE INVENTION

The present invention provides a method and system for modifying the behavior of existing executable code by injecting new code into an executable file. The injection mechanism of the present invention provides techniques for injecting a reference to a new dynamic link library ("DLL") which contains new code into an existing executable file such that, when the code of the executable file is executed, the DLL is automatically loaded and the new code is automatically executed. The injection mechanism provides the automatic loading of the DLL by either modifying a table used by the underlying system to automatically load DLLs or by inserting code that knows how to load the DLL. Thus, a developer desiring to add new behavior to the existing executable code stored in the executable file can do so by providing the new behavior as DLL code. The desired new behavior is preferably provided in an initialization routine of the DLL (e.g., "DLLMain" in the WINDOWS/NT operating system). The injection mechanism ensures that the DLL initialization routine is automatically executed when the DLL is mapped into the executable code image process space and loaded into memory. Thus, using the injection mechanism, any new code can be added to an existing executable file as long as the new code resides in a DLL,. Also, because the DLL is separately testable and modifiable, the injection mechanism of the present invention reduces the time needed to develop and test the new code.

According to the injection mechanism, the reference to the new DLL is injected into the existing executable code in one of two ways. According to the first method, a DLL is injected into existing code by modifying the import table of the existing executable file. An import table is a data structure supported by the underlying operating system that indicates the names of DLLs to be mapped into the executable code when it is run (and loaded if not already loaded into memory when the executable file is loaded). The import table also includes references to the functions within each listed DLL that are called by code in the executable file. (The executable code invokes the functions of a DLL as external references.) The second method for injecting a reference into the executable code modifies the executable code to include DLL loader code, which is provided by the injection mechanism. The DLL loader code relies on the underlying system to provide a mechanism for loading DLLs at run time. This method is useful when no load-time DLL loading mechanism, such as the import table mechanism, is provided by the operating system.

The injection mechanism also provides a technique for injecting security code into the existing executable file to ensure that neither the injected reference to the DLL nor the DLL has been modified. The security code injection technique performs and stores checksums on portions of the executable file and DLL, encrypts a portion of the executable code in the executable file, and inserts security code into the executable file. The security code that is inserted computes checksums on the various portions of the executable file and the DLL and verifies that the checksums are the same as those originally stored. The security code also decrypts and executes the previously encrypted portion of the executable code using an incremental decryption process. The incremental decryption process ensures that a complete version of the unmodified executable file is never visible at any one time. Thus, the injection of security code makes it impossible for somebody to recreate an unmodified version of the existing executable file in a reasonable amount of time.

The injection mechanism is useful in many scenarios. For example, in a globally networked system such as the Internet, licensing code can be incorporated into an existing application and distributed on the system by injecting the licensing code into the application using the injection mechanism. The licensing developer creates a new DLL with the new licensing code accessible through the initialization function of the DLL. The developer then uses the injection mechanism of the present invention to create a modified version of the application that includes a reference to the new DLL. This modified version is then distributed. Further, the newly injected licensing code can be made more

secure by using the injection mechanism to inject security code into the modified application. The injected security code makes it impossible to recreate in a reasonable amount of time an unmodified version of the application that does not include the injected licensing DLL.

The injection mechanism is also useful in other scenarios that require the addition of code to an existing executable file in order to provide modified behavior to existing executable code. As an example, the injection mechanism can be used to modify a network browser, such as an Internet browser, to start and stop applications upon command. In this case, code that starts and stops a designated application is created as a new DLL. The DLL that contains the "start and stop" code is then injected into the browser using the injection mechanism. The application to be started and stopped upon invocation of a command may be designated, for example, by prompting a user for input. Also, the starting and stopping behavior upon command invocation could be provided in the start and stop code using well-known techniques such as a graphical button, menu, or keyboard command.

The injection mechanism also can be used to incorporate additional user interface behavior into an existing application. For example, the injection mechanism could be used to insert a third-party vendor-specific set of menus into an existing application. It is assumed, in this case, that the underlying operating system supports calls to add a menu with menu items into an existing application menu, as well as the ability to handle events caused by the selection of items from the new menu. For example, the MICROSOFT WINDOWS 3.1 operating system provides the "Append Menu," "Insert Menu," and "Set Menu" functions to create and add menus to an application. To add a set of menus, the third-party developer creates new code which creates the menus using the underlying system calls, places the menu appropriately on the screen, and handles any events triggered by the menu items. The newly created menu code is then injected into the application using the injection mechanism.

In a preferred embodiment, the methods and systems of the injection mechanism are implemented on a computer system comprising a central processing unit, a display, a memory, and other input/output devices. Preferred embodiments are designed to operate in an environment that supports shared independent code modules, such as the dynamic link libraries provided by various versions of the WINDOWS operating system. Dynamic link libraries and their use are discussed further in the Charles Petzold, *Programming Windows*, 2d ed., Microsoft Press, Redmond, 1990, pp. 877–915, which is herein incorporated by reference. One skilled in the art will recognize that embodiments of the injection mechanism can be practiced in other environments that support other types of shared, linkable library modules.

FIG. **1** is a block diagram of a general purpose computer system for practicing embodiments of the injection mechanism. The computer system **101** contains a central processing unit (CPU) **102**, a display **103**, a computer memory (memory) **104**, and other input/output devices **105**. The injection mechanism **109** preferably resides in the memory **104** and executes on the CPU **102**. The executable code of an application **106** is shown residing in memory **104** after the injection mechanism **109** has injected a reference to a new DLL **107** and after the injection mechanism **109** has injected security code into the executable file. Other programs **108** also reside in the memory **104**. One skilled in the art will recognize that the preferred injection mechanism can be implemented in a distributed environment where more than one computer system is used to communicate with other computer systems. For example, the application executable code **106** may reside on a different computer system from the DLL **107** or from the injection mechanism **109**. In either case, the injection mechanism **109** preferably relies on the operating system to support the loading of DLLs across different computer systems.

Because the injection mechanism injects a reference to a new DLL and optionally injects security code by adding code into an existing executable file, the injection mechanism needs to have knowledge of the different executable file formats it wishes to manipulate. Although the mechanism itself operates independently of the executable file format, the injection mechanism needs to be aware of the file format in order to determine the proper locations at which references or code should be added. FIG. **2** is a block diagram of a logical format of an executable file that can be used with the present invention. Executable file **201** comprises a header section **205**, an application executable code section **202**, an application import table section **203**, and an application relocatable address table section **204**. The term application is included here for ease of description, although it will be recognized that the executable file may be for some code segment other than an application, for example, a module that comprises part of a program, or a DLL. The header section **205** includes pointers to the application executable code section **202**, the import data section **203**, and the relocatable data section **204**.

The executable file format illustrated in FIG. **2** is a logical representation of the PE file format supported by the MICROSOFT/NT operating system and other operating systems. The particulars of the PE file format are discussed further in *Microsoft Portable Executable and Common Object File Format, Specification* 4.1, Microsoft Corporation, August 1994, which is herein incorporated by reference. One skilled in the art will recognize that this file format is merely illustrative and that other file formats will work. The executable file may be comprised of multiple memory segments, which are not necessarily contiguous. Other figures of the executable file referred to herein are oriented as if they were one logical contiguous sequence of memory addresses. However, it will be appreciated that the layout of these other figures as contiguous is for ease of description purposes. Also, note that although executable file refers to "file " in singular, one skilled in the art would appreciate that the injection mechanism of the present invention would be operable in an environment where multiple files comprise the executable file that is stored in secondary storage.

As discussed above, the injection mechanism injects a reference to a new DLL according to two different methods. The first method modifies the import table of the executable file, whereas the second method modifies the executable file to include DLL loader code that is provided by the injection mechanism. The first method for injecting a reference to a DLL is discussed in detail with reference to FIGS. **3** and **5**. The second method for injecting a reference to a DLL is discussed in detail with reference to FIGS. **6** and **7**.

FIG. **3** is an overview block diagram of the procedure for injecting a reference to a new DLL into an import table of an existing executable file. In FIG. **3**, executable file **301** contains import table **302**. As previously mentioned, the import table **302** enables the underlying operating system to determine which DLLs to map into the process space and to load into memory when the executable file is loaded into memory for execution. Import table **302** contains one import entry per DLL. Each import entry, for example entry **303**, contains the name of the DLL to be loaded and a list of the

external functions defined in the DLL which are referenced by executable file **301**.

To inject new DLL **306**, the injection mechanism finds an appropriate place to add a new entry into import table **302** and adds a new entry, which includes a reference to the injected DLL **306**. Specifically, a new import entry **307** is inserted into the import table and includes the name of the DLL **304** to be injected and a "dummy" function, herein named the stub function **305**. The stub function **305** is not actually referenced by the executable code contained in executable file **301**, but the format of the import table may require the name of at least one function to be included in the entry. As can be seen injected in FIG. **3**, DLL **306** preferably contains a DLL,Main function (the initialization function), which is automatically invoked by the underlying operating system as a result of including the new import entry **307** into import table **302**.

FIG. **4** is a flow diagram of example code that can be placed into an injectable DLL in order to incorporate licensing into an existing application. This licensing code provides an example of code that is added to the DLLMain routine of the injected DLL **306** of FIG. **3** to provide licensing over a network such as the Internet. In step **401**, the licensing code loads any licensing specific data, such as what features of the application are subject to a license, from the application executable file. In step **402**, the code calls some function within a licensing library to determine whether the product is licensed. For example, the licensing library may provide the ability to encrypt a key as a license, and the function referred to in step **402** would then decrypt a stored value and make an assessment as to whether the key is still valid. In step **403**, if it is determined that the product is licensed, then the code continues in step **404**, else continues in step **405**. In step **404**, the code determines whether the license has expired and, if so, continues in step **405**, else returns. In step **405**, the code determines whether the user wishes to properly purchase the product, and if not, terminates the application, else continues in step **406**. In step **406**, the code obtains purchasing information, sends it to the distributor, and then waits to receive a response from the distributor. In step **407**, the code determines whether an error response was received from the distributor and, if so, terminates the application, else continues in step **408**. In step **408**, the code determines whether the user wishes to retry the licensing procedure with the received licensing data from the distributor and, if so, continues back to step **402** to process the data, else terminates the application.

FIG. **5** is a detailed flow diagram of the steps used by the injection mechanism to inject a new DLL using the import table technique. These steps are implemented by the injection mechanism code **109** shown in FIG. **1**. In step **501**, the injection mechanism determines the type of the executable file. Then, in step **502**, if it is a known executable file type, the injection mechanism continues in step **503**, else returns an error. In step **503**, the routine locates where in the executable file the import table for the application is located. For example, according to the executable file format shown in FIG. **2**, the import data ("IData") entry of the header section **205** can be used to locate application import table **203**. Once located, in step **504**, the routine creates a new import entry that refers to the new DLL and adds the new entry to the import table. In step **505**, a reference to the stub function of the new DLL is added to the import entry. Then, in step **506**, the routine adjusts any of the references to relocatable addresses in the executable file that numerically follow the added entry to the import table. This adjustment of relocatable addresses is needed because, by adding a new import table entry, the size of the import table has changed. Thus, everything that was logically below the import table is moved further down. The adjustment of relocatable addresses is similar to the steps performed by a linker/loader mechanism. One such system for adjusting addresses using the PE file format is described in Matt Pietrek, "Peering Inside the PE: A Tour of the Win**32** Portable Executable File Format," *Microsoft Systems Journal*, March, 1994, which is herein incorporated by reference. The injection mechanism of the present invention preferably does not publicize where within the application import table the new entry, which refers to the new DLL, is added. It is not important where the entry is added so long as the step of adjusting relocatable addresses of step **506** is performed appropriately. By not publicizing the location, the amount of time needed to break the security is increased. In step **507**, the routine follows the procedure for injecting security code into the executable file, and then returns. Step **507** is optional, as discussed earlier, and is used to increase the probability that the modified executable file will not be able to be unmodified and subsequently executed without the modifications. The injection of security code is discussed in detail with reference to FIG. **9**.

FIG. **6** is an overview block diagram of the modifications made to an executable file by the injection mechanism to inject a reference to a new DLL using the DLL loader code technique. Executable file **601** represents the logical state of the executable file before any modifications have been made. Executable file **601** contains an indicator **603** of the entry point of the executable code, which is shown located for the purposes of example at address "xxx." Note that the FIG. **6** shows a logical layout of an executable file, in which all the addresses appear to be sequential and continuous. This logical layout is used for the purposes of illustration only as discussed earlier with reference to FIG. **2**. The entry point indicator **603** typically points to the first instruction in the application executable code segment **602**. The instructions comprising executable code segment **602** may vary in size depending upon the instruction set of the underlying computer system.

In FIG. **6**, executable file **604** represents the logical state of the executable file after the injection mechanism has inserted DLL loader code into the executable file **604**. Specifically, the injection mechanism determines a location within the code in which to copy the DLL loader code, copies the DLL loader code, modifies the code entry point indicator **605** (located at address "xxx") to point to the newly added DLL loader code, and stores the value of the previous entry point indicator so that it can be accessed when the DLL loader code is executed. In this manner, when the executable file is executed, it will begin executing at the DLL loader code instead of the code entry point referred to by indicator **605**. The DLL loader code will load the new DLL before executing the original application executable code **606**. The injected DLL loader code **607** contains an instruction at the end of the DLL loader code to transfer control back to the original application executable code **606**. The injected DLL loader code **607** preferably contains a call provided by the underlying operating system to load the new DLL. This call contains a reference to the new DLL **608**, which becomes the injected DLL. As discussed with reference to the import table technique shown in FIG. **3**, the injected DLL **608** contains a DLLMain routine, which is automatically called by the operating system LoadDLL system call and therefore should contain the modifying behavior that the application programmer wishes to add to the executable file. For example, the application programmer could add the licens-

ing procedures discussed earlier or the user interface additions to the DLLMain routine. Injected DLL **608** is presumed to be the same as the DLL injected into the executable file **306** using the import table technique shown in FIG. **3**. Alternatively, the injected DLL loading code could directly invoke a predefined function of the injected DLL.

FIG. **7** is a flow diagram of the steps used by the injection mechanism to inject a new DLL using the DLL loader code technique. In particular, in step **701**, the injection code determines the type of the executable file. In step **702**, if the executable file type is known to the injection code, the routine continues in step **703**, else returns an error because the injection mechanism does not know how to inject code into an unknown executable file type. In step **703**, the routine determines the location of the executable code in the executable file. In step **704**, the routine saves the original application code entry point (e.g., the contents of code entry point indicator **603** in FIG. **6**). In step **705**, the routine adds the DLL loader code to a predetermined location within the located executable code. Similar to the import table technique discussed with reference to FIG. **5**, it is specifically intended that the location where the DLL loader code is placed is not publicized for security reasons. The exact location is preferably immaterial to the operability of the invention. In step **706**, the routine resets the code entry point of the application to the address of the newly added DLL loader code and in step **707** adjusts any relocatable addresses that need adjusting due to the increase in code size at the location where the DLL loader code was added. In step **708**, the injection mechanism code optionally injects security code, and then returns. The injection of security code is discussed in detail with reference to FIG. **9**.

Once the behavioral modifications desired have been added to the executable file by means of injecting a DLL according to either the import table method or the DLL loader method, the executable file when executed will perform any behaviors added to the DLL. However, the modifications made by injecting a DLL are not without security risks. Specifically, without further security measures, a skilled programmer could substitute for the injected DLL another DLL which did not perform the associated behaviors. Or, the programmer could take out the entry in the import table if the import table technique is used. Further, a skilled programmer could modify the injected DLL loader code to load a dummy DLL instead of the injected DLL. For these reasons, the injection mechanism of the present invention provides the added feature of injecting security code into the executable file. Although the techniques used to inject the security code are discussed herein in detail, the particular locations where certain pieces of encrypted code are stored and the particular keys used should preferably not be publicized. These locations and keys are not needed to use or understand the operations of the present invention and are preferably kept secure by the injection mechanism.

FIG. **8**A is a block diagram of the logical layout of an executable file after security code has been injected into the executable file using the import table technique. Executable file **8A01** is shown with import table **8A04** modified to refer to injected DLL **8A02**, as discussed with reference to FIG. **3**. Executable file **8A01** contains a code entry point indicator located at address "xxx," encrypted application code **8A07**, unencrypted application code **8A08**, an import table **8A04**, which refers to injected DLL gA**02**, and injected security code and data **8A05**. The injection mechanism inserts security code and data by adding the appropriate code and data **8A05** to the executable file **8A01** at a predetermined location. The injection mechanism then modifies the application code entry point indicator **8A03** to point to the newly added security code. In addition to injecting security code and data **8AO5**, the injection mechanism encrypts some portion of the original application executable code to further prevent tampering with the modified executable file. In addition, checksums are computed on certain portions of the executable file, to be discussed further below, which are then encrypted and stored as the injected security data shown as part of section **8A05** in FIG. **8**A. Preferably, the injected security code is stored in a predetermined location and the injected encrypted data is stored at another predetermined location, which is preferably not publicized but is kept track of by the injection mechanism. The injection mechanism also inserts a transfer of control instruction back to the application executable code in the security code and data section **8A05**. The location transferred to is preferably the original application executable code entry point combined with the size of the code encrypted starting with the entry point **8A06**. That is, the transfer of control should occur to the first unencrypted location after the original code entry point.

FIG. **8**B is a block diagram of the logical layout of an executable file after security code has been injected into the executable file using the DLL loader code technique. Executable file **8B10** contains a code entry point indicator located at address "xxx," encrypted application code **8B 16**, unencrypted application code **8B17**, injected DLL loader code **8B13**, which refers to an injected DLL **8B11**, and injected security code and data **8B14**. The executable file **8B10** is shown after being modified to include the DLL loader code **8B13** as discussed with reference to FIG. **6**. Thus, the application entry point indicator **8B12** has already been modified to point to the injected DLL loader code **8B13**. Thus, when the executable file is executed, the DLL loader code **8B13** will be executed first. However, in FIG. **6**, the DLL loader code contained a transfer instruction to transfer control back to the original application executable code entry point. In the case shown in FIG. **8**B, where security code will also be injected, this transfer instruction is not added at that point. Instead, after the security code and data **8B14** are injected into the executable file **8B10**, a transfer instruction is added to transfer control back to the original application entry point plus encrypted data **8B15**, which points to the first instruction in the executable code image that appears after the encryption code portion **8B16**. According to this technique, then, the injected security code preferably directly follows the loader code so that it is immediately executed after the injected DLL is loaded by the operating system. However, one skilled in the art will recognize that other techniques are possible instead of depending upon order, including transferring control from the loader code to a predetermined location where the injected security code is stored. In FIG. **8**B as in FIG. **8**A, checksums are computed on various pieces of the executable file and stored at a predetermined location within the executable file **8B10**.

FIG. **9** is an overview flow diagram of the steps performed by the injection mechanism for injecting security code and data into an executable file. The injection mechanism performs and stores checksums and inserts security code into an executable file. Specifically, in step **901**, the injection mechanism performs checksums on various components and saves them in addition to saving the original application code entry point. Preferably, a checksum is computed on the import table and a checksum is computed on a small range of the injected DLL image. So, for example, in the case where a licensing DLL is the injected DLL, some portion of

the licensing code is likely part of the range of the checksum. The small range ensures that the speed of the calculation of checksums is fast, but that the mechanism still accomplishes its security goals. The range of the injected DLL to be checksummed is preferably not publicized and is determined by (and can be changed by) the programmer using the injection mechanism. The checksum operation can be provided by any standard checksumming routine that reads a portion of the data, logically combines the data portion with a mask, and adds the combined data to a compounded checksum result. For example, Table I below provides a sample checksum algorithm:

TABLE 1

```
for each portion of data of a total amount to be checksummed
    x = read (portion of data);
    result = AND (x, mask);
    checksum = checksum + result;
endfor;
```

In step **902**, the routine encrypts the computed checksums and the saved application code entry point and stores the encrypted information in a predetermined location, which also is preferably not disclosed. Note that either the various checksums and the application entry point can be encrypted all at once and decrypted all at once or they can be encrypted and subsequently decrypted using a separate key for each item. In addition, the data can be encrypted and decrypted according to an incremental encryption and decryption technique, which is discussed further below with reference to FIGS. **10**, **11**, **12**, and **14**.

In step **903**, the inject security code routine calls an incremental encryption routine to encrypt portions of the executable code stored in the executable file. The amount of the executable code to be encrypted is preferably not publicized, is small, and can be modified by the programmer using the injection mechanism. A small portion ensures that the speed of the encryption and decryption is fast, but that the mechanism still accomplishes its security goals. In step **904**, the routine determines where the executable code is located within the executable file. In step **905**, the routine adds checksum verification code to a predetermined location within the located executable code. This checksum verification code is stored as part of the injected security code and data **8A05** and **8B14** shown in FIGS. **8**A and **8**B, respectively. In step **906**, the routine adds incremental decryption code to a predetermined location within the located executable code. This incremental decryption code is also part of the injected security code shown as **8A05** and **8B14** in FIGS. **8**A and **8**B, respectively. In step **907**, the injection mechanism adds code to retrieve the encrypted data (which was encrypted in step **902** and stored in the injected security code **8A05** and **8B14** shown in FIGS. **8**A and **8**B), and adds code to transfer control to the saved application entry point taking into account the size of the encrypted portion of the executable code. In step **908**, the routine adjusts any relocatable addresses as necessary due to the addition of the code and data shown as injected security code and data **8A05** and **8B14** in FIGS. **8**A and **8**B, and returns.

Note that, in step **909**, other security checks can be added to the located executable code at some points within this injection procedure. Preferably, security checks, such as making sure the program is running in a particular mode, are added to the inject security code routine in an unpublicized ordering of steps. Thus, the ellipses ". . ." in FIG. **9** represent that step **907** is added somewhere in this process. In a preferred embodiment, the injection mechanism performs a check to make sure the process is not in debug mode and, if so, aborts execution of the executable file. This prevents any undesired viewing of the executable code and security code, which can be used to create an unsecured, or unmodified version of the executable code.

FIG. **10** is a flow diagram of the steps executed by an incremental encryption routine. The incremental encryption routine is invoked, for example in step **903**, as part of the injection of the security code to encrypt portions of the executable code stored within the executable file. In step **1001**, the routine calls a subroutine to determine the size (and location) of the blocks of data that will be encrypted. According to preferred techniques, the encrypted blocks are variable size and the determination routine will compute and store the size for each block. The determination routine is discussed in more detail with reference to FIGS. **11** and **12**. In step **1002**, the routine begins a loop to encrypt the determined number of blocks starting with the first block. In step **1003**, the routine sets the encryption key to a predetermined set of flags (registers) which represent the CPU state as it will exist when the injected incremental decryption code (added in step **906** in FIG. **9**) is executed by the system, i.e., when the executable file is executed.

In particular, the injection mechanism preferably does not publicize the particular flags used as a key. Any flags can serve this purpose as long as they meet the following criteria:

1. The flags do not vary from system to system, or from run of the executable file to run.
2. The flags can be fairly easily predicted.
3. Flags can be chosen based upon a "mode" in which the application will execute.

An example of a flag that does not meet the first criteria is a flag that counts the total number of executions since power-up, because the total number of executions since power-up can vary dramatically from system to system. An example of a flag that does not meet the second criteria is the CPU instruction counter, because where the executable code is physically located in memory when it is loaded will vary. An example of a flag based upon a mode is whether the process is executing in "user mode" or "protected mode" on an Intel processor. Preferably, the injection mechanism does not publicize the exact flags used and how they are combined so that it is more difficult to break into and unmodify the executable file with injected code. Any key that meets these criteria preferably will be operable within the injection mechanism.

In step **1004**, the incremental encryption routine encrypts the current block of data using the determined key for the block size that was specifically determined for that block in step **1001**. Note that any known encryption routine can preferably be used with the injection mechanism of the present invention. In one preferred embodiment, a basic permutation encryption algorithm is utilized. Permutation encryption algorithms, as well as many other types of algorithms, arc described in detail in Bruce Schneier, *Applied Cryptography*, 2d ed, John Wiley & Sons, 1996, which is hereby incorporated by reference. In essence, a permutation encryption routine reorders the bits within the data block being encrypted using a mathematical algorithm that can be duplicated in reverse. Also, for the purposes of the present invention, it is assumed that the encrypted data produced by the chosen algorithm is the exact same size as the original data. However, one skilled in the art, will recognize that any encryption technique can be utilized including those that change the size of the encrypted data from the original data. In that case, the differences in size

must be tracked and accounted for by the injection mechanism, especially in the encryption and decryption routines and in the adjustment of relocatable addresses step performed by many of the routines. In step **1005**, if the encryption algorithm does not add a terminating character to the block encrypted, then the injection mechanism preferably does so. In an alternative embodiment, the encryption and decryption routines keep track of the size of each block and incorporate this size into the decryption procedure. In step **1006**, the incremental encryption routine writes the encrypted block of data into the executable file at the same location where the unencrypted block was. In this way, the injection mechanism replaces the original executable code image with encrypted versions of the image. In step **1007**, the routine checks to see if there are any more blocks that need to be encrypted and, if so, returns to the beginning of the loop at step **1002** to process the next block, else returns.

FIG. **11** is a detailed flow diagram of the steps performed by the injection mechanism to determine the number and size of the encryption blocks of data to be encrypted using the incremental encryption technique. The Determine Encryption Blocks routine takes as input a projected number of blocks and a default size desired for each block. (The default size may also be hardcoded or calculated.) The incremental encryption and decryption process operates on the principle that only a portion of the data to be encrypted should be encrypted or decrypted at any one time so that it is more difficult to break through the encryption. Because the data is preferably never fully decrypted at any one time, it is more difficult for a process to cache a copy of the decrypted code in order to produce an unmodified version of the executable file. Thus, the role of the Determine Encryption Blocks routine is to determine what part of the data to be encrypted will be placed in each encrypted block and to store this information so that the encryption and decryption routines can determine how much data to encrypt/decrypt at any one time.

Also, the Determine Encryption Blocks routine is responsible for setting up each encrypted block to ensure that each block can be decrypted and executed independently from every other block. Thus, the routine constrains the blocks such that there is preferably exactly one entry point into the block from outside the block (a "fall through") and exactly one exit point out of the block to another encrypted code block. Any algorithm capable of maintaining this constraint could be utilized. For example, in the Determine Encryption Blocks routine of FIG. **11**, the routine first scans the non-encrypted execution code and adjusts the portion of the code to be encrypted to ensure that there are no transfers back into encrypted code from outside. Then, the routine divides the data to be encrypted into target blocks and tries to place the maximum amount of data into each target block, up to the default block size, with two exceptions. The exceptions occur when a transfer instruction is encountered. Specifically, when the current source data block contains transfer instructions to target locations within encrypted code that is located outside of the current block, the size of the target block is enlarged to encompass the transfer instruction. Similarly, when the current data block contains a location that is a target of a transfer instruction originating in encrypted code that is outside of the current block, the size of the target block is enlarged to encompass the transfer instruction. These adjustments ensure that, in the decryption process, all instructions that are needed to decrypt a particular portion of code are available.

Specifically, in step **1101**, the routine scans the portion of the source data that is not to be encrypted looking for transfer instructions whose target locations occur within the data to be encrypted. When it encounters such a transfer instruction, the routine adjusts the size of the area of data to be encrypted to stop short of the target location of the transfer instruction. This adjustment ensures that there are no transfers into the encrypted data portion from the unencrypted data portion. The routine also scans the portion of data to be encrypted for transfer instructions with target locations having addresses that occur before the addresses of the corresponding transfer instructions (backward references to encrypted data). When such an instruction is encountered, the routine keeps track of both the target location and the location of the transfer instruction in order to make target block adjustments later (see steps **1106–1108**). In step **1102**, the routine begins a loop to fill target blocks, beginning with the first target block. In one embodiment, the data to be encrypted is copied into a correct size target block in temporary storage. However, one skilled in the art will recognize that other techniques are possible, including those that simply keep track of the original data and the division into different blocks. In step **1103**, the Determine Encryption Block routine sets the size of the current target block equal to the default size. In step **1104**, the routine determines and keeps track of where in the source data the new block begins, as an offset. Assuming the source data begins at the code entry point of the executable file, this offset is the calculation of the start address (the entry point) of the executable code in the executable file plus the sum of the computed sizes of the prior target blocks.

Steps **1105–1110** comprise an inner loop which copies machine instructions to the temporary target block by determining how many can be transferred and whether there are transfer instructions that will affect the size of the current source block. In particular, in step **1105**, the routine reads the next machine instruction in the source block starting with the first instruction. Since instruction sizes can vary, the routine preferably makes sure that it reads the maximum size instruction possible. In step **1106**, the routine determines whether the current instruction is a transfer instruction or the target location of a transfer instruction located outside the current source block and, if so, continues in step **1107**, else continues in step **1109**. In step **1107**, if the current instruction is a transfer instruction, the routine further determines whether the transfer instruction is to a target location in an encrypted block that is outside of the current target block and, if so, continues in step **1108**, else continues in step **1109**. (Transfers to locations within the encrypted block do not cause size adjustments as they do not constitute additional entries to or exits from the current block.) In step **1108**, the routine computes a new size for the current target block to extend to and include the target location of the transfer instruction if the current instruction is a transfer or to extend to and include the corresponding (saved) transfer instruction if the current instruction is the target location of a transfer initiated outside of the current target block, and continues in step **1109**. As mentioned, the purpose of this step is to ensure, for use with the incremental decryption process, that there arc no transfer instructions to an encrypted block outside of the current target block and that there are no transfer instructions from an outside block into the current target block. In step **1109**, the routine copies the current instruction to the current target block in the temporary storage. In step **1110**, the routine determines whether the target block has been filled and if so, continues in step **1111**, else returns to the beginning of the inner loop at step **1105**. In step **1111**, the routine determines whether there are additional source blocks to encrypt and, if so, continues back

to the beginning of the outer loop to determine more target blocks in step **1102**, else returns.

FIG. **12** is an example block diagram of the results of determining the size of encryption blocks according to the technique of FIG. **11**. The original code **1201** is shown on the left hand side and the transferred code **1203** is shown on the right hand side. The transferred code is referred to as "encrypted code," even though it is not encrypted at this point. Referring to FIG. **10**, once the code is divided into blocks and the block sizes are determined, the incremental encryption routine actually encrypts the blocks in step **1004**. Original code **1201** contains a series of instructions shown as beginning at logical address "x." Three target blocks of the encrypted code **1203** are shown as Block **1** (**1204**), Block **2** (**1205**), and Block **3** (**1206**). Using the steps shown in FIG. **11**, the instructions located at logical addresses "x " through logical address "x+60" (instructions **1**–n) are copied directly to Block **1**, because they do not contain transfer instructions and because they exactly fill the default size of a block, which here is assumed to be **64**. The next set of instructions from original code **1201** are continued to be copied into target Block **2** in the encrypted code **1203**. The example shows instructional at logical address "y" being transferred to the beginning of encrypted Block **3**. When the routine reaches the instruction at logical address "y+4," in original code **1201**, the routine determines that the instruction is a transfer instruction, shown here as "jump y+128." Since the calculation of "y+128" is greater than the default size for the block (**64**), it can be seen that this transfer instruction has a target location that is outside of the current target block. It can be further noted that the original code **1201** at logical address "y+128" is a target location that is also encrypted, and therefore Block **3** needs to be extended in size to include the instruction at logical address y+128. Next, at logical address "y+8," the instruction is also a transfer instruction, this time to the instruction of target location logical address "y+136." Again, the original code **1201** at logical address "y+136" is within the code area to be encrypted, and therefore Block **3** must once again be extended in size to include the instruction of the target location "y+136." The routine continues with copying the instructions from original code **1201** into the target block **1203** until it reaches another transfer instruction or until the current size of now enlarged Block **3** is reached. For example, the instructions located through logical address "y+128" are copied over. At logical address "y+132," there is another transfer instruction listed as "jump y+148." This time, the transfer instruction to logical address y+148 transfers to an area of the original code **1201** that is not intended to be encrypted, as shown by the dotted line **1207**. Thus, in this case, Block **3** is not extended in size. Further, since the target block was previously enlarged to include the instruction at location y+136, this instruction is copied over to Block **3**, which terminates the filling of Block **3**.

FIG. **13** is a detailed flow diagram of the verify checksum code added by the injection mechanism to an executable file. Specifically, the inject security code portion of the injection mechanism adds the checksum verification code in step **905** in FIG. **9**. The verify checksum code, when executed by the application, is responsible for retrieving the encrypted security data, setting up the data values appropriately, and verifying that the checksums are correct and that no tampering with the executable file has taken place. In particular, in step **1301**, the verify checksum routine retrieves the encrypted security data block from the predetermined location. As mentioned with reference to FIG. **9**, it is preferable that this location not be publicized, but one skilled in the art will recognize that, once chosen, the routine that generates the original checksum and the routine that verifies the checksum preferably use the same location. In step **1302**, the verify checksum routine decrypts the retrieved security data block. This step assumes, as did FIG. **9**, that all of the security data was encrypted as a single data block. As mentioned, one could have encrypted the security data in individual pieces and the verify checksum routine would need to be changed accordingly. In step **1303**, the routine stores the decrypted checksums and the previously saved application code entry point. In step **1304**, the routine computes the same checksums computed in step **901** of FIG. **9**. These checksums preferably include at least the import table and some range of the injected DLL. In step **1305**, the routine compares the newly generated checksums to the decrypted checksums. Then, in step **1306**, the routine determines whether the checksums are the same, and if so returns (or continues with processing). Otherwise, if the checksums are not the same, the implication is that the executable file has been tampered with, and thus the verify checksum routine returns an error or aborts processing.

FIG. **14** is a detailed flow diagram of the steps performed by incremental decryption. The incremental decryption routine decrypts each block of data previously encrypted and executes each block, one at a time, so that the entire code is never decrypted and executed at once. This procedure helps prevent any kind of illegitimate caching of the executable code to generate an executable file that has not been modified with the injected DLL or security code. Recall that the executable code was encrypted into blocks of varying size, and that each block is guaranteed at this point to contain no transfer instructions to encrypted code outside of the block.

Specifically, in step **1401**, the incremental decryption routines begins a loop over all of the encrypted blocks beginning with the first block. In step **1402**, the routine generates a key using the current designated CPU flags. These flags are the same as discussed relative to the incremental encryption routine of FIG. **10**, and because they do not vary, the key can be determined. In step **1403**, the routine calls a decryption algorithm to decrypt the current block using the determined key. The routine also adds a transfer instruction to transfer to step **1409** (indicated by "LABEL:") so that once the decrypted block is executed, the decrypted code will return back to the incremental decryption routine. This procedure is discussed further below with reference to steps **1407**–**1408**. The decryption routine preferably uses the mirror image of the algorithm used in the incremental encryption routine of FIG. **10**, and any encryption/decryption algorithm that satisfies this criterion should work. In step **1404**, the incremental decryption routine saves the state of the CPU flags in order to later on generate the key for the next block. In step **1405**, the routine restores the saved executable code (application) execution state in order to execute the next block of the application code. This value is initialized to the initial executable state of the code, for example null. In step **1406**, the routine transfers control to the location of the block that was decrypted in step **1403**. The decrypted code block logic is shown in steps **1407**–**1408**. In step **1407** the decrypted code executes, and in step **1408** the transfer instruction to "LABEL:" is executed. This transfers control to step **1409** in the incremental decryption routine. In step **1409**, the routine saves the current application execution state so that it knows what state to restore in step **1405** for execution of the next block of the application code. In step **1410**, the routine restores the CPU flag state that was saved in step **1404** to generate the next key for the next block. In step **1411**, the incremental

decryption routine determines whether there are any more blocks to decrypt and, if so, continues back to the beginning of the loop in step **1401**, else continues in step **1412**. In step **1412**, the routine restores the saved application execution state (it is finished executing all of the encrypted application code) and then continues processing preferably in the application execution code that follows the injected incremental decryption code.

Although the present invention has been described in terms of preferred embodiments, it is not intended that the invention be limited to these embodiments. Equivalent methods, structures, processes, steps, and other modifications within the spirit of the invention fall within the scope of the invention. The scope of the present invention is defined by the claims which follow.

What is claimed:

1. A method in a computer system for modifying an existing executable file so that when the executable file is loaded into memory for execution, control is transferred to an injected dynamic link library prior to transferring control to a main entry point of the executable file, the executable file having an import table indicating each dynamic link library to be mapped and loaded into memory when the executable file is loaded for execution, wherein when a dynamic link library is mapped into memory a main library function of the dynamic link library is executed, the method comprising:

creating the injected dynamic link library with a main library function, the main library function for performing a certain behavior that is not part of the unmodified executable file; and

adding to the import table of the executable file an indication of the injected link library so that when the executable file is loaded into memory control is transferred to the main library function of the dynamic link library to execute the certain behavior prior to transferring control to the main entry point of the executable file.

2. The method of claim **1** wherein the certain behavior is to determine whether the executable file is authorized to execute on the computer system.

3. The method of claim **2** wherein, when the certain behavior determines that the executable file is not authorized to execute on the computer system, the certain behavior terminates execution of the executable file.

4. The method of claim **2** wherein when the certain behavior determines that the executable file is authorized to execute on the computer system, the certain behavior returns from the main library function of the injected dynamic link library so that control can be transferred to the main entry point of the executable file.

5. The method of claim **2**, further comprising adjusting addresses within the executable file to account for the size of the added indication of the injected dynamic link library.

6. A method in a computer system for modifying an executable file so that when the executable file is loaded into memory for execution, control is transferred to an injected dynamic link library prior to transferring control to a main entry point of the executable file, the executable file having an import table indicating each dynamic link library to be mapped and loaded into memory when the executable file is loaded for execution, wherein when a dynamic link library is mapped into memory a main library function of the dynamic link library is executed, the executable file containing a main entry point reference that refers to the main entry point of the executable file, the method comprising:

adding to the import table of the executable file an indication of the injected dynamic link library so that, when the executable file is loaded into memory, control is transferred to the main library function of the injected dynamic link library to execute a certain behavior prior to transferring control to the main entry point of the executable file;

replacing a portion of the executable file with an encrypted version of that portion;

adding to the executable file an encrypted copy of the main entry point reference of the executable file;

adding security code to the executable file; and

setting the main entry point reference of the executable file to refer to the added security code, whereby when the modified executable file is executed, control is transferred to the main library function of the injected dynamic link library and control is then transferred to the added security code referred to by the main entry point reference, wherein the added security code:

determines whether tampering has occurred that affects the execution of the executable file;

when tampering has occurred, terminates execution of the executable file; and

when tampering has not occurred,

replaces the encrypted portion with a decrypted portion; and

transfers control to the main entry point of the executable tile.

7. The method of claim **6**, further comprising adding a checksum for the injected dynamic link library to the executable file, wherein the added security code calculates a checksum for the dynamic link library and compares the calculated checksum to the added checksum to ensure that the dynamic link library has not been modified.

8. The method of claim **6**, further comprising adding a checksum for the import table to the executable file, wherein the added security code calculates a checksum for the import table and compares the calculated checksum to the added checksum to ensure that the import table has not been modified.

9. The method of claim **6** wherein the replacement of the encrypted portion with a decrypted portion uses incremental decryption to decrypt the encrypted portion into subportions and executes each subportion separately.

10. The method of claim **9** wherein the incremental decryption into subportions overwrites a previous subportion with a next subportion before executing the next subportion.

11. The method of claim **9** wherein the added copy of the main entry point reference is encrypted.

12. A method in a computer system for modifying an existing executable file so that when the executable file is loaded into memory for execution, control is transferred to an injected dynamic link library prior to transferring control to a main entry point of the executable file, the executable file containing a main entry point reference that refers to the main entry point, the executable file having executable code, wherein when a dynamic link library is loaded into memory for the executable file, a main library function of the dynamic link library is executed, the method comprising:

locating the executable code in the executable file;

adding loader code to the located executable code, the loader code having instructions for loading the injected dynamic link library into memory;

saving the main entry point referred to by the main entry point reference;

adding transfer of control code into a location that follows the added loader code such that control is transferred to

the saved main entry point after the added loader code is executed; and

setting the main entry point reference to refer to the added loader code.

13. The method of claim **12**, further comprising adjusting addresses within the executable file to account for the sizes of the added loader code and the added transfer of control code.

14. The method of claim **12** wherein a certain behavior is added to the injected dynamic link library to determine whether the executable file is authorized to execute on the computer system.

15. The method of claim **14** wherein, when the certain behavior determines that the executable file is not authorized to execute on the computer system, the certain behavior terminates execution of the executable file.

16. The method of claim **14** wherein, when the certain behavior determines that the executable file is authorized to execute on the computer system, the certain behavior returns from a function of the dynamic link library so that control can be transferred to the main entry point of the executable file.

17. A method in a computer system for modifying an existing executable file to include a reference to new code that contains a certain behavior so that, when the executable file is loaded into memory for execution, control is transferred to the new code with the certain behavior prior to transferring control to a main entry point of the executable file, the executable file containing a main entry point reference that refers to the main entry point, the executable file having executable code, the method comprising:

locating the executable code in the executable file;

adding to the located executable code a reference to the new code with the certain behavior, the reference causing the new code to be executed;

saving the main entry point referred to by the main entry point reference;

adding transfer of control code into a location that follows the added reference to the new code such that control is transferred to the saved main entry point after the new code is executed; and

setting the main entry point reference to refer to the added reference to the new code, so that the new code is executed when the executable file is loaded for execution.

18. The method of claim **17** wherein the adding of the reference to the new code comprises modifying a table within the executable file to include a reference to the new code, the table comprising entries that indicate code to be loaded when the executable file is loaded.

19. The method of claim **17** wherein the adding of the reference to the new code comprises adding loader code within the executable file that loads the new code and transfers execution to a location within the new code.

20. The method of claim **17** wherein the certain behavior of the new code starts and stops another executable code module.

21. The method of claim **17** wherein the certain behavior of the new code adds in a user interface component.

22. The method of claim **18** wherein the new code resides in a dynamic link library.

23. The method of claim **19** wherein the new code resides in a dynamic link library.

24. The method of claim **20** wherein the executable code is a browser application.

25. The method of claim **21** wherein the user interface component is a menu.

26. A method in a computer system for providing a new behavior to executable code stored in an existing executable file, the executable file having an import table indicating each external code library to be mapped and loaded into memory when the executable file is loaded for execution, each external code library having at least one function that can be invoked at runtime by the executable code, wherein when an external code library is mapped and loaded, an initial function within the external library is executed prior to execution of the executable code, the method comprising:

providing a new external code library with an initial function that implements the new behavior;

locating the import table in the executable file; and

adding to the located import table a reference to the provided new external code library, such that, when the executable file is loaded, the initial function of the new external code library is executed, thereby causing the new behavior to be performed.

* * * * *

EXHIBIT I

Asset Purchase Agreement

This Asset Purchase Agreement (hereinafter "Agreement"), is entered into as of November 24, 2003 (the "Effective Date"), by and between Network Commerce Inc., a Chapter 11 debtor in possession, United States Bankruptcy Court for the Western District of Washington (the "Bankruptcy Court") (hereinafter "Seller"), and CRS, LLC, an Washington Limited Liability Company (hereinafter "Purchaser").

Whereas Seller is the owner of full right and title (both legal and equitable) to certain inventions, patents, and applications, defined herein as "Seller Patents";

Whereas, Seller is a party to an action, styled Network Commerce, Inc. v. Microsoft Corporation, Civil Action No. C01-1991P, in the District Court for the Western District of Washington ("the Microsoft Case"); and,

Whereas Purchaser is desirous of acquiring the entire domestic and foreign right title and interest in and to such Seller Patents and all rights of Seller in and to its claims and causes of action in the Microsoft Case.

Now, therefore, Seller and Purchaser hereby covenant and agree as follows:

1. Definitions

1.1. "Seller Patents" shall mean the patents and applications identified on Exhibit A, including (i) all U.S. and foreign patents and patent applications that claim priority to such patents and all U.S. and foreign patents and applications to which such patents and applications relate or claim priority, and (ii) any continuations, continuations-in-part, divisions, reissue applications, extensions, patent Cooperation Treaty applications, or derivatives of any of the foregoing, both foreign and domestic.

1.2. "Prosecution History Files" shall mean all files, documents and tangible things, as those terms have been interpreted pursuant to Federal Rule of Civil Procedure 34, constituting, comprising or relating to investigation, evaluation, prosecution, filing and registration of the Patents, and specifically includes e-mail messages and other electronic or computer stored or generated data.

1.3. "Microsoft Claim" shall mean the claims and causes of action of Seller against Microsoft Corporation which were asserted, or which could have been asserted, by Seller in the Microsoft Case, including (i) all rights to any recoveries of damages, costs or other monetary or other relief, (ii) the right to maintain and prosecute any appeal from any order in the name of the Purchaser or at Purchaser's election in the name of Seller and (iii) the right to maintain and prosecute any further proceedings against Microsoft

in the District Court of the Microsoft Case or in any other court, in the name of the Purchaser or at Purchaser's election in the name of Seller.

1.4. "Net Receipts" shall mean the total recoveries actually received by Purchaser from the exploitation of the Purchased Assets (as defined below in Section 2.3), including all recoveries from licensing, sales, enforcement, lawsuits or settlements, including without limitation, recoveries in the form of cash, stock, personal property, credits or benefits; less all costs and expenses incurred with third parties and not as in-house overhead in connection with prosecuting, licensing, enforcing or defending the Purchased Assets, including without limitation (A) attorneys' and paralegal fees (whether on an hourly or contingent basis and whether for general or local counsel), costs and disbursements, (B) the fees and costs of consultants, experts or technical advisors (other than principals of Purchaser or its affiliates), (C) travel and lodging expenses, (D) duplicating, secretarial, stenographer, postage, courier and similar expenses, (E) filing fees and other Patent Office fees or costs, (F) court costs, (G) legal and other costs related to any re-examination or reissue proceeding or in prosecuting any foreign application, (H) legal and other costs incurred in defending any action or counterclaim in respect of the Patents or the Microsoft Claim, (I) legal and other costs in prosecuting or processing any application, continuing application or continuation in part and (J) patent maintenance fees.

2. Transfer of Rights

2.1. For good and valuable consideration, the receipt and sufficiency of which are hereby acknowledged, Seller hereby agrees to assign and transfer to Purchaser and its representatives, successors and assigns its full and exclusive right, title and interest in and to (i) all Seller Patents, (ii) the Microsoft Claim and (iii) the Microsoft Case. Seller also hereby agrees to assign and transfer to Purchaser and its representatives, successors and assigns its full and exclusive right, title and interest in and to all protectable (e.g., as by patenting) inventions, in the U.S. and every foreign country, described or embodied in the Seller Patents.

2.2. Seller hereby agrees to assign and transfer to Purchaser and its representatives, successors and assigns the full and exclusive right to recover all past damages and other potential relief arising from (i) infringement of the Seller Patents, (ii) the Microsoft Claim and (iii) the Microsoft Case.

2.3. The closing (the "Closing") of the assignment and transfer of the Seller Patents, the Microsoft Claim and other assets described in Section 2.1 and 2.2 (the "Purchased Assets") shall take place on the second business day

following the satisfaction of the conditions set forth in Section 2.4 through 2.6 and Section 4.9 hereof.

2.4. For the purpose of recordation and in accordance with the transfers herein, at the Closing, Seller shall execute separate assignment documents listing the Seller Patents. Upon the written request of the Purchaser and without additional charge or at the Purchaser's expense, the Seller shall execute and deliver to the Purchaser all such additional instruments of transfer, conveyance, endorsement and assignment (in a form satisfactory to the Purchaser) as shall be necessary to transfer the Patents to Purchaser.

2.5. Effective upon the Closing, Seller conveys to Purchaser, its representatives, successors and assigns, the right to make applications on their own behalf for protection of the inventions conveyed herein in the U.S. and foreign countries and to claim, under United States law, the Patent Cooperation Treaty, the International Convention and/or other international arrangements for any such application, priority to any earlier application or patent.

2.6. Within 30 days of the Closing, Seller shall transfer to Purchaser all Prosecution History Files and related files maintained by Seller and its outside and in-house counsel.

3. Payment

3.1. As consideration for the assignment of the Seller Patents and other rights granted by Seller herein, Purchaser shall (i) pay to Seller on or prior to the Closing, the total sum of One Hundred Twenty Thousand U.S. Dollars ($120,000.00) and (ii) grant Seller, its successors and assigns a perpetual right to receive an amount equal to 5% of the Net Receipts (together herein referred to as the "Purchase Price"). Upon the request of Seller, its bankruptcy representative, or its designee, Purchaser and its representatives, successors, and assigns shall provide Seller or its designee with appropriate reports detailing recoveries, costs, and expenses related to Net Receipts, and Seller, its bankruptcy representative, or its designee shall have the right, at its expense, to audit the records pertaining to such recoveries, costs, and expenses.

3.2. Payments under Paragraph 3.1 shall be made by electronic funds transfer. Such payment shall be deemed to be made on the date credited to the following account or to such other account of which Seller may notify Purchaser in writing:

Bank:	Washington Trust Bank
Address:	601 Union Street, Suite 4747

	Seattle, WA 98101
Account Name:	Network Commerce Inc.
ABA Routing #:	125100089
Account #:	2306914618
Bank Contact:	Kirsten Imori

4. Covenants and other Provisions

4.1. Seller represents and warrants that (a) it has the right to assign the Purchased Assets, (b) it is conveying through this Agreement its undivided right, title and interest in and to the Purchased Assets and that no other party has any claim of ownership to the Purchased Assets or any security interest, encumbrance, lien or other claim in or to the Purchased Assets, and (c) no licenses, sublicenses, covenants not to sue or other rights have been granted with respect to the Seller Patents, and no entity has licenses or rights under 11 U.S.C. Section 365(n), except for the licenses and other rights contained in the agreements identified on Exhibit B.

4.2. Seller represents and warrants that no agreements with third parties prevent Seller from entering into this Agreement.

4.3. Seller represents and warrants that, to its knowledge, it has not taken, and will not take, any action materially adversely affecting the validity, enforceability, or issuance of the Seller Patents.

4.4. Seller represents and warrants that none of the Seller Patents is involved in any interference or opposition proceeding and, to Seller's knowledge, no such proceeding is being threatened with respect to any such Seller Patents.

4.5. Seller represents and warrants that subject to appropriate order of the Bankruptcy Court, it is able to convey the Purchased Assets free and clear of any liens, encumbrances, security interests, or other claims (including any claims by Seller's current or former attorneys for fees or costs relating to any matter including the Microsoft Case) to the fullest extent of the Bankruptcy Court's authority to so order, except for the licenses noted on Schedule B.

4.6. Seller shall pay all transfer taxes imposed on the sale of the Purchased Assets, including all sales, gross receipts, excise and gross income taxes.

4.7. Subject to the authority and jurisdiction of the Bankruptcy Court and except as is consistent with the applicable orders of the Bankruptcy Court with respect to the procedures relating to the sale of its assets, Seller covenants and agrees that it shall not execute any writing or do any act whatsoever conflicting with the terms of this Agreement, and that,

following the Closing, Seller will at any time upon request, without further or additional consideration, but at the expense of Purchaser, execute such additional assignments or other writings and perform such additional acts as Purchaser may deem reasonably necessary to perfect Purchaser's ownership of the Purchased Assets. Seller further covenants and agrees, at Purchaser's expense, to render all reasonably necessary assistance following the Closing in making application for, prosecuting in any patent office internationally, and obtaining original, continuation, continuation-in-part, divisional, reissued, reexamined, and national phase patents of the U.S. or of any and all foreign countries on the inventions assigned herein, and in enforcing any rights or choses in action accruing as a result of the rights assigned herein, including enforcing claims in the Microsoft Case and the appeal of the District Court's summary judgment, and by executing statements and other affidavits, it being understood that the foregoing covenant and agreement shall bind, and inure to the benefit of, the assigns and representatives of all parties hereto.

4.8. This Agreement and all matters relating to this Agreement shall be construed and controlled by the laws of the State of Washington. If any legal proceeding or other legal action relating to the Agreement is brought or otherwise initiated, the venue therefore will be the Bankruptcy Court. Purchaser and Seller hereby expressly and irrevocably consent and submit to the jurisdiction of the Bankruptcy Court.

4.9. The Closing and the transactions contemplated herein are and shall be contingent upon (i) the issuance by the Bankruptcy Court of an order, in a form reasonably satisfactory to Purchaser, approving the transactions provided for herein free and clear of liens, encumbrances and rights, to the fullest extent of the Bankruptcy Court's authority to so order (the "Sale Order"); (ii) execution and delivery of the documents and other instruments required to be delivered by Purchaser and Seller on or prior to Closing pursuant to this Agreement; and (iii) receipt by Seller of $120,000.00 (the portion of the Purchase Price to be paid at Closing). The Sale Order shall contain, among other things, a finding that the sale of the Purchased Assets to Purchaser is in good faith within the meaning of Bankruptcy Code Section 363(m).

4.10. Except as otherwise provided in the Agreement, the parties shall pay their respective expenses incurred in connection with the preparation, execution, and delivery of this Agreement and the consummation of the transactions contemplated hereby.

4.11. All notices, requests, demands, and other communications hereunder shall be deemed to have been duly given on the day they are (i) deposited in the U.S. mail, postage prepaid, certified or registered, return receipt requested;

or (ii) sent by air express courier, charges prepaid, and addressed as follows:

4.11.1. If to Purchaser: CRS LLC: Attention Reed Corry, 600 University Street, Suite 2800, Seattle WA 98101

4.11.2. If to Seller: Network Commerce Inc: John R. Knapp, Jr., Cairncross & Hempelmann, P.S., 524 Second Avenue, Suite 500, Seattle, WA 98104-2323.

4.11.3. Such addresses may be changed, from time to time, by means of a written notice delivered by the party seeking to change such address in the manner provided for in this paragraph.

4.12. This Agreement shall be binding upon and inure to the benefit of the parties and their respective successors and assigns.

4.13. This Agreement constitutes the entire agreement between the parties and supersedes all prior agreements and understandings, oral and written, among the undersigned with respect to the subject matter hereof.

In witness whereof, the parties hereto have caused this assignment to be made and executed by duly authorized officers as of the dates indicated below.

Agreed to: **Network Commerce Inc.**	*Agreed to:* **CRS, LLC**
By:_____	By:_____
Name:_____	Name: Reed Corry
Title:_____	Title: Member
Date:_____	Date:_____

or (ii) sent by air express courier, charges prepaid, and addressed as follows:

4.11.1. If to Purchaser: CRS LLC: Attention Reed Corry, 600 University Street, Suite 2800, Seattle WA 98101

4.11.2. If to Seller: Network Commerce Inc: John R. Knapp, Jr., Cairncross & Hempelmann, P.S., 524 Second Avenue, Suite 500, Seattle, WA 98104-2323.

4.11.3. Such addresses may be changed, from time to time, by means of a written notice delivered by the party seeking to change such address in the manner provided for in this paragraph.

4.12. This Agreement shall be binding upon and inure to the benefit of the parties and their respective successors and assigns.

4.13. This Agreement constitutes the entire agreement between the parties and supersedes all prior agreements and understandings, oral and written, among the undersigned with respect to the subject matter hereof.

In witness whereof, the parties hereto have caused this assignment to be made and executed by duly authorized officers as of the dates indicated below.

Agreed to:	*Agreed to:*
Network Commerce Inc.	**CRS, LLC**
By: _____	By: _____
Name: N Scott Dickson	Name: Reed Corry
Title: CFO	Title: Member
Date: 11/24/03	Date: _____

or (ii) sent by air express courier, charges prepaid, and addressed as follows:

4.11.1. If to Purchaser: CRS LLC: Attention Reed Corry, 600 University Street, Suite 2800, Seattle WA 98101

4.11.2. If to Seller: Network Commerce Inc: John R. Knapp, Jr., Cairncross & Hempelmann, P.S., 524 Second Avenue, Suite 500, Seattle, WA 98104-2323.

4.11.3. Such addresses may be changed, from time to time, by means of a written notice delivered by the party seeking to change such address in the manner provided for in this paragraph.

4.12. This Agreement shall be binding upon and inure to the benefit of the parties and their respective successors and assigns.

4.13. This Agreement constitutes the entire agreement between the parties and supersedes all prior agreements and understandings, oral and written, among the undersigned with respect to the subject matter hereof.

In witness whereof, the parties hereto have caused this assignment to be made and executed by duly authorized officers as of the dates indicated below.

Agreed to:	*Agreed to:*
Network Commerce Inc.	**CRS, LLC**
By:_____	By:_____
Name:_____	Name: Reed Corry
Title:_____	Title: Member
Date:_____	Date: 11/24/2003

Exhibit A

Patent 6,073,124
Method and System for Securely Incorporating Electronic Information into an Online Purchasing Application
Date Issued: June 6, 2000

Patent 6,141,698
Method and System for Injecting New Code into Existing Application Code
Date Issued: October 31, 2000

Patent 6,266,681
Method and System for Injecting Code to Conditionally Incorporate A User Interface Component In An HTML Document
Date Issued: April 8, 1999

Patent 6,401,077 B1
Method and System for Providing Additional Behavior Through a Web Page
Date Issued: June 4, 2002

Patent 6,405,316
This is Sub of the '698 patent - Method and System for Injecting New Code into Existing Application Code
Date Issued: June 28, 2000

U.S. Patent Application No. 10/406,624
Date Filed: April 2, 2003

Exhibit B

1. Settlement Agreement, dated March 23, 2001, by and between Network Commerce Inc. and Preview Systems, Inc.

2. Settlement Agreement and General Release, dated June 14, 2002, by and between Network Commerce Inc. and Rainbow Technologies, Inc.

3. Settlement Agreement, dated June 17, 2002, by and between Network Commerce Inc. and Preview Systems, Inc.

4. Settlement Agreement, Mutual Release, License, and Covenant Not to Sue, dated December 17, 2002, by and between Network Commerce Inc. and Aladdin Knowledge Systems, Inc.

5. Settlement Agreement, dated January 27, 2003, by and between Network Commerce Inc. and Liquid Audio, Inc.

EXHIBIT J

Manual of Patent Examining Procedure

Eighth Edition

Incorporating Revision No. 6

THOMSON
WEST

MANUAL OF PATENT EXAMINING PROCEDURE

Eighth Edition

Volume 1

Incorporating Revision No. 6

For those who receive this book as a part of their subscription to Walker on Patents, this volume replaces volume 10 of the third edition set.

THOMSON
WEST

invention claimed in the '283 patent as to the angle limitation and therefore, the '283 patent is not entitled to the filing date of the provisional application under 35 U.S.C. 119(e)(1) and the '283 patent is invalid under 35 U.S.C. 102(b).

A claim is not required in a provisional application. However, for a claim in a later filed nonprovisional application to be entitled to the benefit of the filing date of the provisional application, the written description and drawing(s) (if any) of the provisional application must adequately support and enable the subject matter of the claim in the later filed nonprovisional application. If a claim in the nonprovisional application is not adequately supported by the written description and drawing(s) (if any) of the provisional application (as in *New Railhead*), that claim in the nonprovisional application is not entitled to the benefit of the filing date of the provisional application. If the filing date of the earlier provisional application is necessary, for example, in the case of an interference or to overcome a reference, care must be taken to ensure that the disclosure filed as the provisional application adequately provides (1) a written description of the subject matter of the claim(s) at issue in the later filed nonprovisional application, and (2) an enabling disclosure to permit one of ordinary skill in the art to make and use the claimed invention in the later filed nonprovisional application without undue experimentation.

B. *Claiming the Benefit of Nonprovisional Applications*

The disclosure of a continuation application must be the same as the disclosure of the prior-filed application. See MPEP § 201.07. The disclosure of a divisional application must be the same as the disclosure of the prior-filed application, or include at least that portion of the disclosure of the prior-filed application that is germane to the invention claimed in the divisional application. See MPEP § 201.06. The disclosure of a continuation or divisional application cannot include anything which would constitute new matter if inserted in the prior-filed application. A continuation-in-part application may include matter not disclosed in the prior-filed application. See MPEP § 201.08. Only the claims of the continuation-in-part application that are disclosed in the manner provided by the first paragraph of 35 U.S.C. 112 in the prior-filed application are entitled to the benefit of the filing date of the prior-filed application. If there is a continuous chain of copending nonprovisional applications, each copending application must disclose the claimed invention of the later-filed application in the manner provided by the first paragraph of 35 U.S.C. 112, in order for the later-filed application to be entitled to the benefit of the earliest filing date.

Under 35 U.S.C. 120, a claim in a U.S. application is entitled to the benefit of the filing date of an earlier filed U.S. application if the subject matter of the claim is disclosed in the manner provided by 35 U.S.C. 112, first paragraph, in the earlier filed application. See, e.g., *Tronzo v. Biomet, Inc.*, 156 F.3d 1154, 47 USPQ2d 1829 (Fed. Cir. 1998); *In re Scheiber*, 587 F.2d 59, 199 USPQ 782 (CCPA 1978). A claim in a subsequently filed application that relies on a combination of prior applications may not be entitled to the benefit of an earlier filing date under 35 U.S.C. 120 since 35 U.S.C. 120 requires that the earlier filed application contain a disclosure which complies with 35 U.S.C. 112, first paragraph for each claim in the subsequently filed application. *Studiengesellschaft Kohle m.b.H. v. Shell Oil Co.*, 112 F.3d 1561, 1564, 42 USPQ2d 1674, 1677 (Fed. Cir. 1997).

A claim in the later-filed application is not entitled to the benefit of the filing date of the prior-filed application if the disclosure of the prior-filed application does not enable one skilled in the art to "use" the claimed invention. See *In re Hafner*, 410 F.2d 1403, 1406, 161 USPQ 783, 786 (CCPA 1969) ("[T]o be entitled to the benefits provided by [35 U.S.C. 120], the invention disclosed in the "previously filed" application must be described therein in such a manner as to satisfy *all* the requirements of the first paragraph of [35 U.S.C.] 112, including that which requires the description to be sufficient to enable one skilled in the art to *use* the [invention].").

Where the prior application (a nonprovisional application) is found to be fatally defective because of insufficient disclosure to support allowable claims, a later-filed application filed as a "continuation-in-part" of the first application to supply the deficiency is not entitled to the benefit of the filing date of the first application. *Hunt Co. v. Mallinckrodt Chemical Works*, 177 F.2d 583, 587, 83 USPQ 277, 281 (2d Cir. 1949) and cases cited therein.

Any claim in a continuation-in-part application which is directed *solely* to subject matter adequately disclosed under 35 U.S.C. 112 in the parent nonprovisional application is entitled to the benefit of the filing date of the parent nonprovisional application. However, if a claim in a continuation-in-part application recites a feature which was not disclosed or adequately supported by a proper disclosure under 35 U.S.C. 112 in the parent nonprovisional application, but which was first introduced or adequately supported in the continuation-in-part application, such a claim is entitled only to the filing date of the continuation-in-part application; *In re Chu*, 66 F.3d 292, 36 USPQ2d 1089 (Fed. Cir. 1995); *Transco Products, Inc. v. Performance Contracting Inc.*, 38 F.3d 551, 32 USPQ2d 1077 (Fed. Cir. 1994); *In re Van Lagenhoven*, 458 F.2d 132, 136, 173 USPQ 426, 429 (CCPA 1972); and *Chromalloy American Corp. v. Alloy Surfaces Co., Inc.*, 339 F. Supp. 859, 874, 173 USPQ 295, 306 (D. Del. 1972).

By way of further illustration, if the claims of a continuation-in-part application which are only entitled to the continuation-in-part filing date "read on" published, publicly used or sold, or patented subject matter (e.g., as in a genus-species relationship) a rejection under 35 U.S.C. 102 would be proper. Cases of interest in this regard are as follows: *Mendenhall v. Cedarapids Inc.*, 5 F.3d 1557, 28 USPQ2d 1081 (Fed. Cir. 1993); *In re Lukach*, 442 F.2d 967, 169 USPQ 795 (CCPA 1971); *In re Hafner*, 410 F.2d 1403, 161 USPQ 783 (CCPA 1969); *In re Ruscetta*, 255 F.2d 687, 118 USPQ 101 (CCPA 1958); *In re Steenbock*, 83 F.2d 912, 30 USPQ 45 (CCPA 1936); and *Ex parte Hageman*, 179 USPQ 747 (Bd. App. 1971).

C. *Form Paragraphs*

Form paragraphs 2.09 and 2.10 should be used where the claims of the later-filed application are not adequately disclosed or enabled by the disclosure of the prior application.

**>

¶ *2.09 Heading for Conditions for Benefit Claims Under 35 U.S.C. 119(e), 120, 121, or 365(c)*

Applicant's claim for the benefit of a prior-filed application under 35 U.S.C. 119(e) or under 35 U.S.C. 120, 121, or 365(c) is acknowledged. Applicant has not complied with one or more conditions for receiving the benefit of an earlier filing date under 35 U.S.C. **[1]** as follows:

Examiner Note:

1. In bracket 1, insert either or both --119(e)-- or --120--.
2. One or more of form paragraphs 2.10 to 2.11.01 or 2.38 to 2.40 must follow depending upon the circumstances.

<

¶ *2.10 Disclosure of Prior-Filed Application Does Not Provide Support for Claimed Subject Matter*

The later-filed application must be an application for a patent for an invention which is also disclosed in the prior application (the parent or original nonprovisional application or provisional application). The disclosure of the invention in the parent application and in the later-filed application must be sufficient to comply with the requirements of the first paragraph of 35 U.S.C. 112. See *Transco Products, Inc. v. Performance Contracting, Inc.*, 38 F.3d 551, 32 USPQ2d 1077 (Fed. Cir. 1994).

The disclosure of the prior-filed application, Application No. **[1]**, fails to provide adequate support or enablement in the manner provided by the first paragraph of 35 U.S.C. 112 for one or more claims of this application. **[2]**

Examiner Note:

1. This form paragraph must be preceded by heading form paragraph 2.09.
2. This form paragraph may be used when there is lack of support or enablement in the prior-filed application for the claims in the application that is claiming the benefit of the prior-filed application under 35 U.S.C. 120, 121, or 365(c) or under 35 U.S.C. 119(e). The prior-filed application can be a provisional application or a nonprovisional application.
3. In bracket 1, insert the application number of the prior-filed application.
4. In bracket 2, provide an explanation of lack of support or enablement. If only some of the claims are not entitled to the benefit of the filing date of the prior application, the examiner should include a list those claims after the explanation (e.g., "Accordingly, claims 1-10 are not entitled to the benefit of the prior application.").

Form paragraph 2.10.01 should be used where applicant is claiming the benefit of a prior nonprovisional application under 35 U.S.C. 120, 121, or 365(c) and the relationship (continuation or divisional) of the applications should be changed to continuation-in-part because the disclosure of the later-filed application contains matter not disclosed in the prior-filed nonprovisional application.

¶ *2.10.01 Continuation or Divisional Application Contains New Matter Relative to the Prior-Filed Application*

Applicant states that this application is a continuation or divisional application of the prior-filed application. A continuation or divisional application cannot include new matter. Applicant is required to change the relationship (continuation or divisional

EXHIBIT K

(31)

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

3COM CORPORATION,

Plaintiff

v.

D-LINK SYSTEMS, INC.,

Defendant.

C.A. No. 03-014 GMS

MEMORANDUM AND ORDER

I. INTRODUCTION

On January 7, 2003, the plaintiff, 3Com Corporation ("3Com") filed the instant action alleging infringement of three patents relating to network interface adapters. The defendant, D-Link Systems, Inc. ("D-Link"), moves to transfer this case to the United States District Court for the Northern District of California pursuant to 28 U.S.C. § 1404(a) (D.I. 11). For the following reasons, the court will grant the defendant's motion.

II. DISCUSSION

D-Link moves to transfer this action to the District Court for the Northern District of California pursuant to 28 U.S.C. § 1404(a). Section 1404(a) provides that "[f]or convenience of [the] parties and witnesses, in the interest of justice," the court may transfer a civil action "to any other district . . . where it might have been brought." 28 U.S.C. § 1404(a). It is the movant's burden to establish the need for transfer, and 'the plaintiff's choice of venue [will] not be lightly disturbed.' *Jumara v. State Farm Ins. Co.*, 55 F.3d 873, 879 (3d Cir. 1995) (citations omitted).

When considering a motion to transfer, the court must determine 'whether on balance the

litigation would more conveniently proceed and the interest of justice be better served by transfer to a different forum.' *Id.*. This inquiry requires "a multi-factor balancing test" embracing not only the statutory criteria of convenience of the parties and the witnesses and the interest of justice, but all relevant factors, including certain private and public interests. *Id.* at 875, 879. These private interests include the plaintiff's choice of forum; the defendant's preference; whether the claim arose elsewhere; and the location of books and record, to the extent that they could not be produced in the alternative forum.[1] *Id.* at 879. Among the relevant public interests are: "[t]he enforceability of the judgment; practical considerations that could make the trial easy, expeditious, or inexpensive; the relative administrative difficulty in the two fora resulting from court congestion; the local interest in deciding local controversies at home; [and] the public policies of the fora." *Id.* at 879-80 (citations omitted).

Upon consideration of these factors, the court finds that D-Link has met its burden of demonstrating that transfer is appropriate. First, it is clear that this case could have been brought in the Northern District of California. Any federal district court possesses subject matter jurisdiction over federal patent law claims such as those at issue in the present action. 28 U.S.C. §§ 1331 and 1338. Further, venue is proper in the Northern District of California because the defendant is a California corporation with its sole place of business in that state. 28 U.S.C. § 1400(b) ("Any civil action for patent infringement may be brought in the judicial district where the defendant resides . . . .").

---

[1] The first three of these private interest collapse into other portions of the *Jumara* analysis. The court, therefore, will consider them in the context of the entire inquiry only. *See Affymetrix, Inc. v. Synteni, Inc. and Incite Pharmaceuticals, Inc.*, 28 F. Supp. 2d 192 (D. Del. 1998).

Having determined that the case could be properly heard in the Northern District of California, the court now considers whether it would more conveniently proceed in that forum and whether the interest of justice supports a transfer to that district. Again, the court finds that these criteria are met. First, the court notes that although 3Com is a Delaware corporation, its principal place of business is in Santa Clara, California. D-Link is a California corporation with its sole place of business in Irvine, California. Although some of D-Link's products, including the accused products, are sold in Delaware, the connection to this forum ends there. Neither 3Com nor D-link maintains or owns any facility, property, or personnel in Delaware. Instead, the headquarters of both parties are located in California. Neither party has any books, records, or other documents in this district. Apparently, none of the acts related to the development of the accused products occurred in this district, while many, if not all, of these acts occurred in California. Clearly, litigating this case there would cause less disruption to business operations of each corporation, while eliminating the cost and time of cross-country transportation of persons and documents. In addition, D-Link was forced to retain local counsel for purposes of litigating in this district. Were the case transferred to California, this additional expense would not be required.

In addition, none of the anticipated third-party witnesses is subject to compulsory process in Delaware, but they may be compelled to testify in the Northern District of California. These witnesses include individuals involved in the development of the accused products, such as employees of the manufacturers of the products, Realtek Semiconductor Corp. ("Realtek") and Via Technologies, Inc. ("Via"). Realtek and Via are Taiwanese companies with offices and/or agents in northern California. Futhermore, at least one of the inventors of the accused products and one of the prosecuting attorneys could not be compelled to testify in this court. By contrast, each appears

to live in northern California, and would be subject to compulsory process there. Finally, at least two witnesses with knowledge of allegedly invalidating prior art are subject to compulsory process in northern California, but not Delaware. Even if these witnesses were willing to travel to Delaware to testify in this court, it is certainly very inconvenient for them to do so, especially compared to traveling to a court in the state of their residence and employment. Convenience, cost, and expediency, then, favor a transfer.

The remaining factors of court congestion, the enforceability of the judgment, and the public polices of the fora neither favor nor counsel against transfer. These factors remain neutral in the court's analysis.

III. CONCLUSION

In short, Delaware seems to have little interest in the present dispute between these parties, while justice, convenience, cost, and expediency favor a forum in California. The court recognizes that the Northern District of California is not the plaintiff's choice of forum for the present action; however, it is an exceedingly more convenient and appropriate forum than Delaware. In other words, the movant has shown that 'the litigation would more conveniently proceed and the interest of justice be better served by transfer' to California. *Jumara*, 55 F.3d at 879 (citations omitted). As such, transfer is appropriate.

For the aforementioned reasons, IT IS HEREBY ORDERED that:

1. The defendant's Motion to Transfer the case to the United States District Court for the Northern District of California (D.I. 11) is GRANTED.

Dated: April 25, 2003

UNITED STATES DISTRICT JUDGE

EXHIBIT L

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

GREEN ISLE PARTNERS, LTD., S.E.,	:	
	:	
Plaintiff,	:	
	:	
v.	:	Civil Action No. 01-202-JJF
	:	
THE RITZ-CARLTON HOTEL CO., L.L.C., et al.,	:	
	:	
Defendants.	:	

MEMORANDUM ORDER

Presently before the Court is Defendants' Motion To Dismiss, Or In The Alternative, To Stay Or Transfer. (D.I.18). For the reasons discussed, the Court will grant Defendants' Motion To Transfer Under 28 U.S.C. §1404(a).[1] (D.I.18).

I. BACKGROUND

Plaintiff, Green Isle Partners, Ltd. S.E., ("Green Isle") is a Florida limited partnership. (D.I.1). The Ritz-Carlton Hotel Company, L.L.C, and The Ritz-Carlton Hotel Company of Puerto Rico, Inc., (collectively "Ritz-Carlton Defendants") are Delaware corporations, headquartered in Georgia. (D.I.27 at 4). Marriott International Inc., Marriott Distribution Services, Inc., and Marriott International Capital Corp., (collectively "Marriott Defendants") are also incorporated in Delaware with their principal place of business in Maryland. Id.

[1]Because the Court will transfer this action, it will not consider Defendant's Motion To Dismiss Or In The Alternative, To Stay, as these motions are best addressed by the United States District Court for the District of Puerto Rico.

On March 30, 2001, Green Isle filed an eleven-count Complaint, arising from the development and operations of its Ritz-Carlton, San Juan Hotel, Spa & Casino in Puerto Rico ("Hotel"), against the Ritz-Carlton Defendants, the Marriott Defendants, and Avendra, L.L.C. ("Avendra"), (collectively "Defendants"). (D.I.1). This is the third litigation commenced concerning the Hotel. (D.I.27 at 5). In October 2000, Green Isle commenced an action in the Delaware Chancery Court against the Ritz-Carlton Defendants seeking the production of the Ritz-Carlton Defendants' books and records relating to the Hotel, and in November 2000, the Ritz-Carlton Defendants filed an action against Green Isle in the Courts of Puerto Rico alleging breach of contract, unjust enrichment, promissory estoppel, and negligent misrepresentation. Id. Additionally, Green Isle has filed for bankruptcy in the United States District Court for the Southern District of Florida under Chapter 11 of the Bankruptcy Code. (D.I.27 at 5-6).

On May 25, 2001, Defendants filed the instant Motion To Dismiss, Or In The Alternative, To Stay Or Transfer. (D.I.19).

II. DISCUSSION

Transfer of a civil action is governed by 28 U.S.C. §1404(a) which provides, "[f]or the convenience of parties and witnesses, in the interest of justice, a district court may transfer any civil action to any other district or division where it might

have been brought." The thought behind §1404(a) is that when a civil action "presents issues and requires witnesses that make one District Court more convenient than another, the trial judge can, after findings, transfer the whole action to the more convenient court." White v. ABCO Engineering Corp., 199 F.3d 140, 143 (3d Cir. 1999) citing Continental Grain Co. v. Barge FBL-585, 364 U.S. 19 (1960).

The United States Court of Appeals for the Third Circuit has instructed that when reviewing a motion to transfer under 28 U.S.C. §1404(a) district courts must consider a number of public and private interests. Specifically, the private interests to be considered are

> (1) the plaintiff's choice of forum, (2) the defendant's preferred forum, (3) whether the claim arose elsewhere, (4) the convenience of the parties due to their relative physical and financial conditions, (5) the convenience of the expected witnesses, but only so far as the witnesses might be unavailable for trial if the trial is conducted in a certain forum, and (5) the location of books and records, to the extent that these books and records could not be produced in a certain forum.

Jumara v. State Farm Ins. Co., 55 F.3d 873, 879 (3d Cir. 1995). The public interests are

> (1) the enforceability of the judgment, (2) practical considerations regarding the ease, speed, or expense of trial, (3) the administrative difficulty due to court congestion, (4) the local interest in deciding local controversies in the home forum, (5) the public policies of the two fora, and (6) the trial judge's familiarity with the applicable state law in diversity cases.

Id. When determining whether or not transfer is warranted in the circumstances presented, district courts must balance all of the relevant factors. Id. at 883. The burden is upon the movant to establish that the balance of the interests strongly weighs in favor of transfer, and a transfer will be denied if the factors are evenly balanced or weigh only slightly in favor of the transfer. See Continental Cas. Co. v. American Home Assurance Co., 61 F.Supp. 2d 128, 131 (D.Del. 1999). Because it is undisputed that Plaintiff could have brought the instant action in the District of Puerto Rico, the Court's only task is to determine whether the factors enumerated in § 1404(a) and by the Third Circuit, warrant a transfer.

At the outset, Green Isle contends that its choice of forum is paramount, and should not be disturbed. (D.I.27 at 36). The Defendants contend that Green Isle's choice of forum is less important because Green Isle has no connection to Delaware. (D.I.19 at 36).

The Court recognizes that the plaintiff's choice of forum is entitled to substantial deference and should not be lightly disturbed. Shutte v. Armco Steel Corp., 431 F.2d 22, 25 (3d Cir. 1920). However, when the plaintiff's choice of forum is not its "home turf,"[2] the transfer of a case will generally be regarded

[2] "Home turf" is defined as the forum closest to the plaintiff's residence or principal place of business which can exercise personal jurisdiction over the defendants. Continental

as less inconvenient to a plaintiff if the plaintiff has not chosen its home turf or a forum where the alleged wrongful activity occurred. In re ML-Lee Acquisition Fund II, L.P., 816 F.Supp. 973, 976 (D.Del. 1993). Another factor in the balance of convenience is whether the plaintiff has offered any "substantive reasons ... indicating that the convenience to it of litigating in [this forum] even approaches the inconvenience which trial in this forum will impose on the defendants and their witness." Continental Cas. Co., 61 F.Supp. 2d at 131 quoting Clopay Corp. v. Newell Companies, Inc., 527 F.Supp 733, 737 (D.Del. 1981).[3]

A. PRIVATE INTERESTS[4]

1. CONVENIENCE OF THE PARTIES

Green Isle contends that this district is convenient for Defendants because the Marriott Defendants and the Ritz-Carlton Defendants are incorporated in Delaware. (D.I.27 at 36-37). Moreover, Green Isle contends that Defendants are wealthy,

---

Cas. Co. v. American Home Assurance Co., 61 F.Supp. 2d 128, 131 n5 (D.Del. 1999).

[3]Green Isle chose to file this action in Delaware because it is "where all the Defendants are incorporated and because all the defendants subjected themselves to suit in Delaware." (D.I.27 at 36).

[4]The Court will not separately consider the first three private interests enumerated by the Third Circuit, specifically, the plaintiff's choice of forum, the defendant's choice of forum, and whether the claim arose elsewhere, because the Court concludes that consideration of such factors are subsumed in an examination of the remaining factors. See Continental Casualty Co., 61 F.Supp.2d at 131 n7.

multinational corporations in the travel industry, and therefore, are capable of financing a trial in Delaware. Id. at 37. Defendants respond that Puerto Rico would be a more convenient forum for all parties because the parties have extensive contacts with Puerto Rico and have demonstrated the ability to litigate in that forum. (D.I.19 at 36-37). Specifically, Defendants contend that Puerto Rico would be convenient for Green Isle because it has previously litigated in Puerto Rico and maintains outside counsel in Puerto Rico. Id. at 37.

The Court acknowledges that a defendant who chooses to incorporate in Delaware should not be heard to complain when another entity has chosen to sue it in Delaware. However, where a defendant can demonstrate that an alternative forum would be more convenient and would better serve the interests of justice, because that forum has substantial connections with the litigation, incorporation in Delaware will not prevent transfer. See SAS of Puerto Rico, Inc. v. Puerto Rico Telephone, Co., 833 F.Supp. 450 (D.Del. 1993). In the instant case, the Court concludes that incorporation in Delaware will not preclude the transfer of this action, because, as discussed, the Defendants have demonstrated that the District of Puerto Rico would be a more convenient forum and has substantial connections to the litigation.

2. CONVENIENCE OF THE WITNESSES

Defendants have identified several nonparty witnesses[5] who would be called to testify regarding the operation of the Hotel and the Hotel's local purchasing and distribution practices, but are not subject to compulsory process in this District. (D.I.19 at 38-39). In response, Green Isle has not identified any witnesses who would be inconvenienced by a trial in Puerto Rico. (D.I.27). Therefore, the Court concludes that this factor weighs heavily in favor of transfer.

3. LOCATION OF BOOKS AND RECORDS

Defendants contend that all the sources of proof are located in Puerto Rico, and thus this factor should weigh in favor of transfer. (D.I.19 at 38-39). Green Isle responds that the location of books and records should be irrelevant in light of the current technology. (D.I.27 at 37-38). While the Court recognizes that current technology has reduced the burden of litigating in distant forums, it is still an inconvenience. The Court finds that it would be more convenient for the parties to litigate in Puerto Rico, where all or most of the sources of proof are located. Therefore, the Court concludes that this factor weighs slightly in favor of transfer.

---

[5]Ken DeStefano, former Green Isle on-site representative, Dilio Mena, former Director of Finance at the Hotel, Eric Rodriguez, former Casino Manager and employee of the Hotel and several local vendors. (D.I.19 at 38-39).

B. PUBLIC FACTORS[6]

1. PRACTICAL CONSIDERATIONS THAT COULD MAKE TRIAL EASY, EXPEDITIOUS, OR INEXPENSIVE

Defendants do not offer practical considerations that could make trial easy, expeditious, or inexpensive in Puerto Rico other than those already discussed, i.e. nonparty witnesses located in Puerto Rico, books and records located in Puerto Rico, and other substantial connections to Puerto Rico. Similarly, Green Isle does not offer any practical considerations that would make trial in Delaware easy, expeditious, or inexpensive. Therefore, to avoid double-counting, the Court will assign this factor no weight.

2. ADMINISTRATIVE DIFFICULTIES POSED BY COURT CONGESTION

The Court's own research reveals that the median times to civil trial in Puerto Rico and Delaware differ by only two months. Federal Court Management Statistics 2000, available at http://www.uscourts.gov/cgi-bin/cmsd2000.pl. Therefore, the Court assigns no weight to this factor.

3. LOCAL INTEREST IN RESOLVING LOCAL CONTROVERSIES AT HOME

Defendants contend that Puerto Rico has an interest in resolving litigation surrounding real property located in Puerto

---

[6]The Court will not address the enforceability of a judgment and the public policies of the fora as these factors are not contested by the parties and not relevant to the instant case.

Rico. (D.I.19 at 40). Green Isle responds that Puerto Rico does not have a local interest because the cause of action arose, not in Puerto Rico, but in Georgia, Maryland, or Florida. (D.I.27 at 34-35). Because the Court is not examining the merits of the Complaint, it is unclear where the cause of action arose. Therefore, it is unclear in whose favor this factor should weigh. However, it is clear that this factor does not weigh in favor of this litigation continuing in Delaware.

4. THE TRIAL JUDGE'S FAMILIARITY WITH THE APPLICABLE STATE LAW

The parties dispute what law, Puerto Rico, Georgia, or Florida, will apply to Green Isle's common law causes of action. (D.I.19 at 30; D.I.27 at 31). Again, because the Court is not examining the merits of the Complaint, the Court cannot make a determination of what law will apply. Accordingly, it is unclear in whose favor this factor should weigh. However, the parties agree that Delaware law will not apply to the common law causes of action, and therefore, this factor does not weigh in favor of this litigation continuing in Delaware. Id.

B. SUMMARY

The Court concludes that a balancing of the private interests heavily favors transferring this case to the District of Puerto Rico. The potential unavailability of Defendants' nonparty witnesses for trial is a substantial factor in the Court's decision to transfer. Furthermore, litigating in Puerto

Rico, where the sources of proof are located, will clearly be more convenient for all parties.

While the public interests appear to be in equipoise, Green Isle has not established a substantial connection to this forum, in any of the public interests, that is sufficient to overcome the private interests that weigh in favor of transfer. Therefore, under the facts of this case, the Defendants Delaware incorporation, alone, is not enough to maintain this action in this forum.

NOW THEREFORE, IT IS HEREBY ORDERED this 2 day of November that:

(1) Defendants' Motion To Transfer Pursuant To 28 U.S.C. § 1404(a)(D.I.18) is **GRANTED**.

(2) This action shall be transferred to The United States District Court for the District of Puerto Rico.

UNITED STATES DISTRICT JUDGE

EXHIBIT M

IN THE UNITED STATES DISTRICT COURT

FOR THE DISTRICT OF DELAWARE

ALTERA CORPORATION,	:	
	:	
Plaintiff and Counterdefendant,	:	
	:	
v.	:	Civil Action No. 95-242-JJF
	:	
XILINX, INC.,	:	
	:	
Defendant and Counterclaimant.	:	

Bruce M. Stargatt, Josy W. Ingersoll, James P. Hughes, Jr., of YOUNG, CONAWAY, STARGATT & TAYLOR, Wilmington, Delaware. Herbert F. Schwartz, Robert J. Goldman, Douglas J. Gilbert, of FISH & NEAVE, New York, New York. Attorneys for Plaintiff.

Douglas E. Whitney, Lisa B. Baeurle, of MORRIS, NICHOLS, ARSHT & TUNNELL, Wilmington, Delaware. Alan H. MacPherson, Joseph A. Greco, Kenneth E. Leeds, Scott R. Brown, Peter H. Kang, of SKJERVEN, MORRILL, MACPHERSON, FRANKLIN & FRIEL, San Jose, California. Attorneys for Defendant.

MEMORANDUM OPINION

March 29, 1996

Wilmington, Delaware

FARNAN, District Judge.

I. INTRODUCTION

Presently before the court is the Motion to Transfer (D.I. 9) of Defendant Xilinx, Inc. ("Xilinx") pursuant to 29 U.S.C. § 1404(a). Xilinx asks the Court to transfer the present case to the United States District Court for the Northern District of California. (D.I. 10).[1] Judge Eugene Lynch of that district already has before him two lawsuits between the parties, one filed by Xilinx and the other filed by Altera Corporation ("Altera"). Xilinx claims that the convenience of the parties and interests of justice direct that the present lawsuit should also be before the same judge. (D.I. 10 at 4-6). Altera disputes the transfer, claiming that these same factors mandate that the lawsuit remain in this district.

The Court agrees with Xilinx that the convenience of the parties and interests of justice are better served by the transfer of the case to the Northern District of California, and will grant Xilinx's Motion to Transfer to that district.

II. BACKGROUND

Altera is a California corporation with its principal place of business in San Jose, California, and was founded in 1983. Xilinx is a Delaware Corporation with its principal place of business in San Jose, California, and was founded in 1984.

---

1. Originally, Xilinx wanted the cases transferred to Judge Aguilar in the San Jose Division of the Northern District of California, but they have since been transferred to Judge Eugene Lynch in San Francisco.

Both companies manufacture integrated circuits known as "programmable logic devices," which a customer can configure through software programs to perform a variety of logic functions. Many of the devices can be configured to change the logic functions originally programmed into the chip. Until 1992, Altera made Erasable Programmable Logic Devices ("EPLDs") based on Erasable Programmable Read Only Memory ("EPROM"), while Xilinx made Field Programmable Gate Arrays ("FPGAs") based on Static Random Access Memory ("SRAM").

In the fall of 1992, Altera developed the "FLEX 8000" family, a line of products using FPGA based on SRAM, the device and memory systems heretofore used by Xilinx. Xilinx notified Altera that the new line infringed its patents. In turn, Altera notified Xilinx that Xilinx products infringed Altera's patents, although Altera had never accused Xilinx of patent infringement when these products had issued and been patented.

Xilinx sued Altera for patent infringement in the district court in San Jose, California, and later that day, Altera sued Xilinx in the same court against Xilinx. The district court joined the cases for discovery purposes, but decided that the trials would proceed separately, and the Xilinx's lawsuit would be tried first.

On April 20, 1995, Altera brought suit in this Court, claiming that Xilinx's "next generation" products infringed its patents. On May 30, 1995, Xilinx counterclaimed, claiming that Altera products infringed Xilinx's patents.

III. DISCUSSION

Section 1404(a) vests district courts with broad discretion in deciding whether to transfer a case to another district. It provides that "[f]or the convenience of parties and witnesses, in the interest of justice, a district court may transfer any civil action to any other district or division where it may have been brought." 28 U.S.C. § 1404(a). While a plaintiff's choice of a proper forum is a paramount consideration in determining a transfer request, where the convenience of the parties or the interests of justice is strongly in favor of the defendant, the plaintiff's choice will not prevail. Shutte v. Armco Steel Corp., 431 F.2d 22, 25 (3d Cir. 1970), cert. denied, 401 U.S. 910 (1971) (citations omitted).

After weighing the assertions and arguments of both parties, the Court concludes that the interests of justice and convenience of the parties outweigh the plaintiff's choice of forum. As a threshold matter, there is no dispute that the action could have been brought in the Northern District of California, since two other lawsuits involving the same parties have been brought there.

The balance of the convenience of the parties tilts toward California, because the headquarters of both parties are located in Northern California, as is their principle places of business. Moreover, most of the documents and witnesses relevant to the litigation are located in California.

Moreover, judicial economy is better served by transferring this case to the Northern District of California. The district court in California is familiar with the complex technologies, product structures and prior art involved as well as at least one of the patents.[2] While the Court is mindful of Altera's concern that its case may not be heard in Northern California until 1997 because of the civil per-judge caseload, the Court believes that the Northern District of California is fully capable of efficiently handling this patent case.

IV. CONCLUSION

For the reasons discussed, the Court will grant Xilinx' Motion to Transfer to the Northern District of California.

An appropriate Order will follow.

---

2. Altera contends that only one patent overlaps, and that is the patent at issue in Xilinx's counterclaim. (See D.I.27 at 29 & n.21.)

IN THE UNITED STATES DISTRICT COURT

FOR THE DISTRICT OF DELAWARE

ALTERA CORPORATION,	:	
	:	
Plaintiff and Counterdefendant,	:	
	:	
v.	:	Civil Action No. 95-242-JJF
	:	
XILINX, INC.,	:	
	:	
Defendant and Counterclaimant.	:	

O R D E R

At Wilmington this 29 day of March, 1996, for the reasons set forth in the Memorandum Opinion issued this date,

IT IS HEREBY ORDERED that Xilinx' Motion to Transfer to the Northern District of California (D.I. 9) is GRANTED.

Joseph J. Farnan Jr.
UNITED STATES DISTRICT JUDGE

EXHIBIT N

IN THE UNITED STATES DISTRICT COURT

FOR THE DISTRICT OF DELAWARE

CONOPCO, INC. d/b/a UNIPATH DIAGNOSTICS COMPANY,	:	
Plaintiff,	:	
v.	:	Civil Action No. 01-308-JJF
PFIZER INC., and PRINCETON BIOMEDITECH CORPORATION,	:	
Defendants.	:	

MEMORANDUM ORDER

Presently before me is a Joint Motion To Transfer Venue Pursuant To 28 U.S.C. § 1404(a) filed by Pfizer Inc. ("Pfizer") and Princeton Biomeditech Corporation ("PBM") (collectively "Defendants"). (D.I.20). For the reasons discussed, I will grant Defendants' Motion and transfer the instant action to the United States District Court for the District of New Jersey. (D.I.20).

I. BACKGROUND

Prior to this litigation, Unipath Diagnostics Corporation ("Unipath") filed suit against PBM and Warner-Lambert Company (now Pfizer), separately, in the United States District Court for the District of New Jersey alleging infringement of U.S. Patent Nos. 5,622,871, 5,602,040, and 5,656,503 (collectively "the parent patents"). (D.I.21 at 6). Both cases were assigned to the Honorable Katharine S. Hayden. (D.I.21 at 6). In each case, Judge Hayden granted summary judgment of noninfringement. (D.I.22 Exhibits A and B). Presently, the decisions are on appeal before

§ 1404(a) district courts must consider, among other things, private[2] and public[3] interests. See Jumara v. State Farm Ins. Co., 55 F.3d 873 (3d Cir. 1995). When determining whether or not transfer is warranted in the circumstances presented, district courts must balance all of the relevant factors. Id. at 883. The burden is upon the movant to establish that the balance of the interests strongly weighs in favor of transfer, and a transfer will be denied if the factors are evenly balanced or weigh only slightly in favor of the transfer. See Continental Cas. Co. v. American Home Assurance Co., 61 F.Supp. 2d 128, 131 (D.Del. 1999).

---

[2]The private interests are:

(1) the plaintiff's choice of forum, (2) the defendant's preferred forum, (3) whether the claim arose elsewhere, (4) the convenience of the parties due to their relative physical and financial conditions, (5) the convenience of the expected witnesses, but only so far as the witnesses might be unavailable for trial if the trial is conducted in a certain forum, and (5) the location of books and records, to the extent that these books and records could not be produced in a certain forum.

Jumara v. State Farm Ins. Co., 55 F.3d 873, 883 (3d Cir. 1995).

[3]The public interests are:

(1) the enforceability of the judgment, (2) practical considerations regarding the ease, speed, or expense of trial, (3) the administrative difficulty due to court congestion, (4) the local interest in deciding local controversies in the home forum, (5) the public policies of the two fora, and (6) the trial judge's familiarity with the applicable state law in diversity cases.

Id.

would be assigned the instant action under New Jersey Local Rule of Civil Procedure 40.1, is uniquely familiar with the technology, the parent patents, and the accused products. Judge Hayden's previous rulings on the parent patents, enable her to confront any new issues that might arise under the derivative '660 patent without any duplication of effort.

Therefore, I conclude that a transfer to the District of New Jersey will prevent the potential for inconsistent rulings and is the best use of limited judicial resources. Further, I am persuaded that to allow this action to remain in Delaware would validate Plaintiff's apparent forum shopping to avoid unfavorable rulings in the District of New Jersey.

NOW THEREFORE, IT IS HEREBY ORDERED this 15 day of November that:

(1) Defendants' Joint Motion To Transfer Venue Pursuant To 28 U.S.C. § 1404(a)(D.I.20) is **GRANTED**.

(2) This action is transferred to The United States District Court for the District of New Jersey.

Joseph J. Farnan, Jr.
UNITED STATES DISTRICT JUDGE